UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

INSTITUTO DE MATEMÁTICA

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

VITOR MARQUES DE MIRANDA

Projeto Porting

RIO DE JANEIRO

2019

VITOR MARQUES DE MIRANDA

Projeto Porting

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profa. Valeria Menezes Bastos

RIO DE JANEIRO

2019

CIP - Catalogação na Publicação

M672p

Miranda, Vitor Marques de
Projeto Porting / Vitor Marques de Miranda. -- Rio de Janeiro, 2019.
130 f.

Orientadora: Valéria Menezes Bastos.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2019.

1. Portabilidade de aplicações. 2. Automação. 3. Wine. I. Bastos, Valéria Menezes, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

VITOR MARQUES DE MIRANDA

Projeto Porting

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em: ___ de _____ de _____.

BANCA EXAMINADORA:

_____
Prof. Valeria Menezes Bastos
(Presidente)

_____
Prof. Miguel Jonathan

_____
Prof. Nelson Quilula Vasconcellos

RESUMO

O foco do Projeto Porting foi a pesquisa ao redor dos diferentes métodos para realizar a execução de aplicações do sistema operacional Windows no ambiente macOS, com o intuito de criar uma aplicação para automatizar o processo de instalação, de forma que o usuário não precise de conhecimentos avançados para realizá-la. O projeto culminou na criação da aplicação Porting Kit, que é o grande foco deste trabalho.

Palavras-chave: Portabilidade. Wine. Automação.

ABSTRACT

The focus of the Porting Project was the research of the different methods to execute an application made for the Windows operating system in the macOS environment, in order to create an application to automate the installation process, so that the user would not need advanced knowledge to realize it. The project culminated in the creation of the Porting Kit application, which is the main focus of this work.

Keywords: Portability. Wine. Automation.

LISTA DE FIGURAS

Figura 1:	Logo oficial “Powered by Cider” (Movido a Cider)	20
Figura 2:	Logos oficiais do CrossOver Games e do CrossOver (também usado no CrossOver Pro/Office)	20
Figura 3:	Antigo logo oficial do Wineskin	22
Figura 4:	Logo do CXZ na comunidade Porting Team	23
Figura 5:	Logo oficial do Bordeaux for Linux / Mac OS X	23
Figura 6:	Logo oficial do CXEx	24
Figura 7:	Logo oficial do CiderX	24
Figura 8:	Logo oficial do Wineskin Pro	25
Figura 9:	Atual logo oficial do Wineskin	27
Figura 10:	Arquitetura do Wineskin	30
Figura 11:	Fluxograma do usuário avançado no Porting Kit	31
Figura 12:	Fluxograma do usuário leigo no Porting Kit	31
Figura 13:	Ícone utilizado no Appleverse Wizard	32
Figura 14:	Janela do Appleverse Wizard em sua versão final	32
Figura 15:	Ícone utilizado no Wigidrasil	33
Figura 16:	Última janela do Wigidrasil / Primeira janela do Gamma Studio	34
Figura 17:	Evolução do ícone do Gamma Studio	34
Figura 18:	Ícone utilizado no Gamma Board	35
Figura 19:	Novo ícones do Porting Center e do Porting Kit respectivamente	37
Figura 20:	Janela do Gamma Studio abrindo um port	42
Figura 21:	Janela de exportação avançada de WSI no Porting Kit	43
Figura 22:	Ícone de “The Elder Scrolls IV: Oblivion” antes e depois da conversão pelo img2icns	44

Figura 23: Ícone do programa Pré-Visualização 44

Figura 24: Exemplo do resultado da extração de ícone de um arquivo de vídeo 45

Figura 25: Ícones obtidos a partir de uma pasta comum, do diretório raiz (HD) e de um diretório Home 45

Figura 26: Exemplo de uma imagem SVG após a rasterização 45

Figura 27: Exemplo de extração de ICO do EXE instalador do jogo “Marvel Heroes" 46

Figura 28: Exemplo de extração de PNG do ICO extraído do executável do jogo "Mass Effect" 46

Figura 29: Ícone de um documento do Word 47

Figura 30: Trecho faltante para transformar um ICON em um arquivo ICO 48

Figura 31: Listagem dos ícones presentes no maior ICO do jogo “Mass Effect”. Vários ícones, um ICO 49

Figura 32: Ícone extraído pelo icotool e ícone visualizado pelo sistema de “Robust MotionDeblur” 50

Figura 33: O ícone de “Commandos: Behind Enemy Lines” original, aprimorado com Pré-Visualização e com Gimp, respectivamente 51

Figura 34: Ícone original baseado em Bomberman e ícone aprimorado pelo SmillaEnlarger, respectivamente 52

Figura 35: Ícone de "Commandos: Behind Enemy Lines” aprimorado sem contorno vazio 53

Figura 36: Ícone de "Commandos: Behind Enemy Lines” aprimorado com contorno vazio 53

Figura 37: Ícone da empresa Lionhead Studios extraído do jogo "Black & White" 54

Figura 38: Imagem SVG de logotipo da companhia Apple após rasterização 55

Figura 39: QuickLook de port do jogo Phantasmat 58

Figura 40: Tela de extração de dados de EXE do jogo “The Elder Scrolls IV: Oblivion” 60

Figura 41: Janela original de descoberta de bibliotecas/winetricks necessários 62

Figura 42: Nova janela com Descrição e Winetricks por Biblioteca 62

Figura 43: Aba de Drives no Porting Center 67

Figura 44: Janela de troca de tipo de drive no Porting Center 68

Figura 45: Aba de Drives no Porting Kit 68

Figura 46: Aba de MenuBar no Porting Center 69

Figura 47: Janela com comandos úteis para botões do menu no Porting Center 69

Figura 48: Aba de Tela do Porting Center 70

Figura 49: Aba de Tela do Porting Kit 71

Figura 50: Aba de Port no Porting Center 72

Figura 51: Aba de EXEs no Porting Center 74

Figura 52: Aba de Extensões no Porting Center 75

Figura 53: Janela de instalação de winetricks do Porting Center 77

Figura 54: Visualização de log do Wine no Porting Kit 78

Figura 55: Layout final do Porting Center 82

Figura 56: Página de busca do Porting Kit 84

Figura 57: Página do jogo “The Elder Scrolls V: Skyrim” 85

Figura 58: Nova página de busca do Porting Kit 85

Figura 59: Nova página do jogo “2 Fast Driver” 86

Figura 60: Primeiro layout do Porting Kit; antes a Biblioteca era a totalidade do app 88

Figura 61: Sketch original da aba da Biblioteca (aba Local) 89

Figura 62: Layout baseado no sketch original 90

Figura 63: Layout originalmente proposto por Ahmed para o Porting Kit 90

Figura 64: Layout final de Ahmed para o Porting Kit 91

Figura 65: Layout do Porting Kit adaptado para janelas redimensionáveis 91

Figura 66: Layout da biblioteca do Porting Kit na versão 2.1.0 93

Figura 67: Descrição de uma aplicação 93

Figura 68: Menu de clique direito para um port local e para um port no servidor na versão 2.1.0 94

Figura 69: Nova janela com a área de progresso 96

Figura 70: Package de instalação do MySQL Community 96

Figura 71: Janela de instalação the “The Elder Scrolls IV - Oblivion” no Porting Kit 97

Figura 72: Janela de votação 99

Figura 73: Aba para edição da descrição 100

Figura 74: Aba de Configurações 101

Figura 75: Sketch original da aba de Rankings 102

Figura 76: Sketch original da aba de Comunidade 103

Figura 77: Aba de Comunidade em sua versão final 104

Figura 78: Aba de FAQ, com as perguntas consideradas pertinentes 105

Figura 79: Janela de especificações aberta ao se pressionar o botão 106

Figura 80: Quantidade de instalações bem sucedidas via Porting Kit de acordo com o Google Analytics 118

LISTA DE TABELAS

Tabela 1: Motores CXZ/CXEx e seus correspondentes 26

Tabela 2: Os diferentes tipos de port 28

Tabela 3: Comparação das ferramentas usadas para rasterizar arquivos SVG 56

Tabela 4: Lista dos identificadores e placeholders por informação 59

SUMÁRIO

1. INTRODUÇÃO **12**

2. OBJETIVO **15**

3. APLICABILIDADE **16**

3.1 - CYTOCLUS 17

3.2 - OUTROS FEEDBACKS 18

4. TIPOS DE PORT **19**

5. FUNCIONAMENTO **29**

5.1 - WINESKIN 30

5.2 - PORTING KIT 30

6. EVOLUÇÃO **32**

6.1 - PORTING CENTER 39

6.1.1 - Janela Principal **40**

6.1.2 - Obtenção de ícone **43**

6.1.2.1 - Extração de ícone de .EXE 46

6.1.2.2 - Extração de .ICO 49

6.1.2.3 - Aprimoramento de Imagem 51

6.1.2.4 - Rasterização de .SVG 54

6.1.3 - Obtenção de Dados de .EXE **57**

6.1.3.1 - Obtenção de Nome/Versão/Copyright 57

6.1.3.2 - Descobrir bibliotecas necessárias 60

6.1.4 - Obtenção de Caminho **63**

6.1.4.1 - Extração de .LNK 64

6.1.4.2 - Extração de .DESKTOP 64

6.1.4.3 - Extração de .INF 66

6.1.5 - Controle de Drives **66**

6.1.6 - Edição de Menu **68**

6.1.7 - Configuração de Tela **70**

6.1.8 - Detalhes de Port **72**

6.1.9 - Gerenciamento de EXEs **73**

6.1.10 - Gerenciamento de Extensões **74**

6.1.11 - Instalação de Winetricks **76**

6.1.12 - Os Arquivos WSI **78**

6.1.13 - Múltiplos Idiomas **79**

6.1.14 - Manuseio de WSIs **80**

6.2 - PORTING KIT 87

6.2.1 - Novidades **97**

6.2.2 - Biblioteca **98**

6.2.2.1 - Propriedades 100
6.2.3 - Rankings 101
6.2.4 - Comunidade 102
6.2.5 - Ajuda e Doar 105
6.2.6 - Criação de Wrapper com WSI 107
6.2.6.1 - Instalação de client silenciosa 107
6.2.6.2 - Verificação de legitimidade 110
6.2.6.3 - Atualização de Componentes 111
6.2.6.4 - Construindo o Wrapper 111
6.2.6.5 - Instalação de Winetricks Silenciosa 112
6.2.6.6 - Configurando o Caminho de Execução 113
6.2.6.7 - Reparo para Placas de Vídeo 114
6.2.6.8 - Configurações Finais 115
7. CONCLUSÃO 116
7.1 - UPLOAD DE PORTS COM PAUL THE TALL 117
7.2 - RECEPÇÃO DO PÚBLICO 117
8. REFERÊNCIAS 119
9. GLOSSÁRIO 126
10. ANEXOS 128

1. INTRODUÇÃO

Em 2009, quando obtive meu MacBook, um notebook da marca Apple, uma de minhas primeiras observações foi o fato de que os programas para atividades padrões (como editar um texto) eram diferentes, mas logo percebi também que faltavam certos programas dos quais eu precisava.

No início, procurei por programas alternativos que fossem desenvolvidos para o *Mac*, mas me deparei com alguns casos mais complicados, como quando precisei realizar a cópia de um DVD de vídeo, e vi que o *Toaster*, o programa mais utilizado para este fim, era pago, enquanto no Windows o *DVD Shrink* era gratuito.

Além da questão do preço, eu já estava acostumado com a interface do DVD Shrink, e eu teria que me adaptar a interface do Toaster, além de pagar por um recurso que eu sempre considerei básico. Pouco depois, vi que isso não afetava apenas os programas utilitários, e que o campo mais sofredor era o de jogos.

A grande maioria dos jogos não possui versão para *Mac OS X*, o que para mim era algo tão sério quanto a falta do *DVD Shrink* (como grande fã de jogos). Para contornar o problema, comecei a procurar soluções pela internet a fora e a realizar diferentes testes.

Primeiramente, inspirado pelos meus tempos de estudo no CEFET/RJ Uned-NI, em que todos os computadores possuíam máquinas virtuais com *Linux* e *Windows*, criei uma máquina virtual de *Windows* com o ***VirtualBox*** para instalar os programas e jogos que precisava.

Os resultados foram desanimadores, visto que a maioria dos jogos não funcionavam, e que apesar da grande maioria dos programas funcionarem, eu precisava carregar um segundo sistema operacional para usá-los, o que acarretava muito tempo, além de resultar em um grande consumo de memória e baixo desempenho.

Sem desanimar, continuei a procurar uma solução, e me deparei com o ***Wine***. O *Wine* é um programa desenvolvido para *Linux* e *Mac*, que permite utilizar programas de *Windows* sem precisar passar por um processo de emulação, repassando as chamadas dos aplicativos de *Windows* para bibliotecas do sistema

nativo, e traduzindo todo o resto para POSIX, transformando-o em um “programa semi-nativo”.

As razões pelas quais o programa não poderia ser considerado nativo eram estéticas e funcionais. O *Wine* possui diversos *motores* (engines) que fazem seus aplicativos funcionarem, mas ele só pode possuir um motor por computador, o que complica muito o processo. Além disso, os aplicativos instalados via *Wine* não têm uma aparência nativa, pois fica muito claro que eles estão sendo executados por meio do próprio *Wine*.

Sem tirar o *Wine* da mente, busquei por algum programa que fosse similar, e descobri então o ***WineBottler***. O *WineBottler* é similar ao *Wine* e resolvia a questão da aparência nativa, pelo menos parcialmente. Infelizmente, o problema do motor único também ocorria nele. Cada aplicação funciona de maneira diferente, por isso certas aplicações podem funcionar perfeitamente com um motor e não abrir com outro.

O *Wine* não havia me levado a nada, já o WineBottler havia me permitido instalar o *DVD Shrink* e *Capitão Claw* (um jogo antigo que joguei na época), mas eu mantinha as esperanças de encontrar um método melhor. Depois de um tempo, descobri o ***CrossOver Games***.

O CrossOver Games é pago, mas ainda assim possuía resultados excelentes, visual nativo, e dispensava o uso de vários motores já que executava a instalação das demais bibliotecas necessárias para o funcionamento da aplicação desejada (caso a mesma estivesse presente em sua base de dados). Infelizmente havia dois detalhes: além de ser pago, o CrossOver Games precisava estar instalado para que os programas pudessem ser executados, e seu motor não parecia ser perfeito para todos os casos.

Depois de pesquisar um pouco mais, cheguei ao ponto principal de minha pesquisa: ***Wineskin***. Com o Wineskin era possível instalar aplicativos com múltiplos motores, rodando com aparência nativa, e não era preciso ter ele instalado para que os programas funcionassem.

Parecia ser perfeito, até que eu me deparei com a grande desvantagem dele: a sua interface. A janela do utilitário do Wineskin é incompreensível para um

usuário inexperiente, o que por sorte não era o meu caso. Depois de um tempo, me adaptei a usar a janela, mas sem me agradar muito com a idéia.

Pouco tempo depois, descobri uma variação do Wineskin chamada **CXZ**, mas ela precisava estar instalada no sistema para que os aplicativos instalados por meio dela funcionassem (mesmo problema do CrossOver Games). Quando o Wineskin foi atualizado, surgiram brevemente mais duas variações do mesmo, o **Novo CXZ** e o **CXEx**, logo em seguida. Nenhum deles era superior ao Wineskin, exceto no critério de economia do espaço em disco, o qual no meu caso era irrelevante.

Depois de um tempo usando o mesmo, e tendo instalado diversas aplicações (principalmente jogos), pensei em criar um programa que pudesse armazenar essas instruções, e que então pudesse realizá-las sozinho; em outras palavras, um programa que pudesse instalar aplicações de Windows automaticamente, sem que o usuário precisasse conhecer o Wineskin. Essa idéia foi o que deu início ao projeto.

2. OBJETIVO

A meta principal do projeto é construir uma aplicação (Porting Kit) que permita instalar e executar programas feitos para o sistema operacional Windows no sistema operacional macOS, sem grandes dificuldades.

Para que isso seja atingido sem a necessidade de virtualização, a aplicação utilizará o Wineskin como motor, tornando-o simplificado, de maneira que o usuário não necessitará de conhecimentos avançados para realizar o objetivo proposto, partindo do pressuposto que outra pessoa já realizou a instalação desse programa no macOS antes usando o próprio Wineskin.

Isso se deve ao fato que, por meio do Porting Kit, um usuário pode converter um port (aplicação de Windows instalada no macOS via Wineskin) em um arquivo WSI, o qual é basicamente composto por um conjunto de instruções que ensina o próprio Porting Kit a criar um réplica do port original. Isso por sua vez se faz necessário pois: levar uma aplicação não assinada (como é o caso de port) de um computador para outro causa alertas de segurança no sistema; ports podem ser bem grandes dependendo da aplicação portada; e em alguns casos, transferir um programa de uma máquina para outra pode ser considerado pirataria. Todos esses problemas são evitados ao usar o arquivo WSI para recriar o port.

3. APLICABILIDADE

Uma dúvida comum seria: quais tipos de aplicações necessitam dessa metodologia para serem instaladas no macOS? Em que caso não haveria outra alternativa mais adequada, como recompilar a aplicação para o sistema desejado? Algumas dessas circunstâncias serão enumeradas abaixo.

O primeiro caso a mencionar é o mais óbvio: aplicações de código fechado que não são multiplataforma. Como exemplo é possível citar o DVD Shrink, referenciado na introdução. Alguns programas simplesmente não possuem substitutos adequados, e por isso precisam ser portados para que possam ser utilizados.

Em seguida vale mencionar as aplicações institucionais. Quando aplicações são desenvolvidas no meio acadêmico elas por diversas vezes podem possuir apenas uma versão para Windows, o que inibe os usuários de Mac de tirar proveito das mesmas. Neste caso, a simples geração de um arquivo WSI que possa instalar essa aplicação, disponibilizado no mesmo domínio que a mesma, seria mais que o suficiente para tornar usuários de Mac capazes de instalá-la.

Além da comodidade do usuário ter de baixar apenas um pequeno arquivo a mais, existem vantagens para o próprio desenvolvedor, tais como economia com hospedagem (caso a opção anteriormente adotada fosse disponibilizar um port completo para download), ter de se preocupar apenas com a versão para Windows quando tiver de corrigir *bugs*, não ter de fornecer alertas sobre desativar opções de segurança para utilizar o app (pois esta alteração não é necessária para o Porting Kit), entre outras coisas mais.

Por possuírem fins bens específicos, essas aplicações podem não possuir substitutos. O Cytoclus é um desses programas, e será mencionado mais adiante neste capítulo.

E por fim: os jogos; os quais tiveram uma grande ênfase no desenvolvimento deste projeto. Apesar de existirem vários jogos para o sistema macOS, eles se enquadram em um tipo de aplicação em que há alternativas, mas os usuários não as desejam; eles querem a aplicação original.

Poderia-se cobrar das empresas pequenas-médias para que seus jogos fossem portados e colocados no Porting Kit. Considerando o valor médio cobrado por PaulTheTall para realizar um port (US$ 10.000), poderia se cobrar o mesmo.

Com o Porting Kit, qualquer um com um pouco mais de conhecimento tem poder para criar seu próprio arquivo WSI, no entanto o Porting Kit só reconhece os arquivos como confiáveis (sem dar alertas) se estes são obtidos do servidor, evitando assim a possibilidade de fraude e mantendo o usuário alerta. Isso tem o potencial para estender o mercado de Porting de app de Windows para Mac, tornando-o mais simples e mais acessível.

3.1 - CYTOCLUS

O CytoClus foi o programa testado com Porting Center / Kit para avaliar o seu uso dentro da universidade, para provar que não seria muito difícil criar um port com o uso do Porting Center, e muito menos realizar a instalação do mesmo com o Porting Kit.

A pedidos da Professora Myrian Christina de Aragão Costa, D.Sc., a qual possuía apenas um Mac a sua disposição e necessitava do Windows apenas para utilizar este programa, concedeu uma cópia do instalador do programa para que o mesmo fosse portado.

Logo ao tentar iniciar ao instalador do programa houve um erro. O erro alertava sobre a ausência de uma dependência necessária para o programa ser instalado: o .NET 3.5. Na época ele ainda não era bem suportado pelo Wine, portando foi necessário aguardar um pouco.

Posteriormente, em uma nova tentativa, tentou-se realizar a instalação do winetrick deste pacote (*dotnet35*) em um port. Em uma saída de erro bem clara deixada durante a instalação do winetrick, se viu que ele precisava do winetrick *msxml3* para ser executado.

O wrapper foi criado do zero, e então foram instalados o *msxml3* e o *dotnet35*. A instalação foi bem sucedida, e o programa executou perfeitamente nos testes iniciais. Para criar o WSI do mesmo foi necessário definir um caminho relativo

para o arquivo executável da aplicação, pois este caminho variava de instalação para instalação.

Por fim, o port do CytoClus foi instalado e utilizado com sucesso pela professora.

3.2 - OUTROS FEEDBACKS

O Porting Kit também foi testado pelo aluno Renan Barbieri, o que forneceu o seu feedback da usabilidade da aplicação. A interface foi elogiada, assim como a facilidade na criação de port, sendo até mesmo considerada mais simples do que no Windows.

Também não foi observada nenhuma lentidão durante a execução de um jogos disponibilizados na base do programa. Características que foram apontadas para melhoras futuras são o fato da aplicação ter muitos jogos disponíveis, e isso passar o entendimento de que o Porting Kit seja apenas para esse tipo de programa.

Outra coisa que aparenta ser pouco clara é a questão do licenciamento das aplicações. Um indivíduo pode apenas instalar um programa pelo Porting Kit se uma possuir uma licença válida ou cópia legítima do mesmo. A aplicação não fornece aplicações distribuídas de maneira ilegal, mas esse dado não fica tão exposto quanto deveria.

4. TIPOS DE PORT

Depois de manusear por muito tempo diferentes tipos de ports iniciei uma extensa pesquisa pela história do port de aplicações de *Windows* para *Mac OS X*, e ao fim descobri diversos dados curiosos, além de ter desvendado muitas coisas em relação à evolução dos ports.

Antes de começar, é importante lembrar que neste capítulo estarei citando a evolução dos ports apenas, e não de outros métodos como máquinas virtuais e dual boots. Quaisquer outros métodos mencionados neste capítulo foram feitos por empresas que posteriormente entraram neste núcleo, e estão aqui apenas para mostrar a evolução da empresa em termos de instalação de programas de Windows no Mac OS X.

Além disso, mais um conceito precisa ser explicado: para ser compatível com múltiplos sistemas operacionais, o Wine foi originalmente escrito de maneira que todas as suas janelas eram desenhadas no OS X usando apenas o *X Windows System*, também conhecido como *X11* (isso mudou apenas anos depois, com a chegada do *Mac Driver*). Os sistemas 10.5 a 10.7 da Apple foram lançados com o *X11* já instalado, mas as versões posteriores passaram a precisar do *XQuartz* instalado, uma versão em código-aberto do *X11*. As aplicações da CrossOver sempre possuíram seu próprio *build* do *X11* embutido.

Partindo do princípio, o primeiro tipo de port lançado a público foi o *Cider*, lançado pela companhia *TransGaming* em Outubro de 2006, pouco depois da *Apple* adotar processadores *Intel* em seus computadores. O *Cider* era uma aplicação de uso empresarial, ou seja: uma empresa desenvolvia uma aplicação para *Windows* e pagava a *TransGaming* para *portar* sua aplicação para *Mac*; esta então desenvolvia um port *Cider* específico para aquela aplicação, o qual era então distribuído pela empresa que desenvolveu a aplicação original.

Figura 1: Logo oficial "Powered by Cider" (Movido a Cider)

Enquanto isso, a companhia *CodeWeavers* era proprietária da aplicação *CrossOver*, a qual era capaz de instalar aplicações de *Windows* no *Linux* por meio de seu próprio *build* do *Wine*. Em 9 de janeiro de 2007, a *CodeWeavers* lançou sua aplicação para *Mac*, e em 25 de março de 2008 dividiram o *CrossOver* em ambos *Linux* e *Mac* em dois programas: *CrossOver Games* e *CrossOver Pro*, os quais tinham focos em jogos e em softwares em geral, respectivamente. Estas mesmas aplicações vieram a se fundir novamente formando um novo *CrossOver* em 6 de março de 2012.

Figura 2: Logos oficiais do CrossOver Games e do CrossOver (também usado no CrossOver Pro/Office)

Não se tem certeza de quando ao certo esse movimento começou, no entanto, em determinado momento do ano de 2008, um grupo de usuários de *Mac* começou a remover os programas de dentro dos ports *Cider* e a compartilhá-los na internet, indagando-se se não seria possível instalar outros programas com aqueles mesmos wrappers, focando-se principalmente em jogos. Esse movimento ocorreu na comunidade iBrain (http://ibrain.com.ua/), a qual era uma comunidade voltada a

auxiliar usuário de produtos Apple, como iPods, iPhones e Macs, onde os usuários partilhavam suas experiências para conseguir resolver problemas.

Em 15 de setembro de 2008, a *CodeWeavers*, em uma tentativa de se auto-promover, criou um port do navegador de código aberto *Chromium* para que o mesmo rodasse no *Mac* usando a tecnologia dos motores *CrossOver*, no entanto que não necessitava da aplicação *CrossOver* (a qual era paga) para ser executado.

Em 17 de março de 2009, precisamente às 17:51, foi anunciado um subdomínio na iBrain chamado *The Game Porting Team* (http://dev.ibrain.com.ua/), onde os usuários poderiam dividir suas experiências no desenvolvimento de ports.

Levou um certo tempo para que a comunidade de *porters* se desse conta da existência do port da *CodeWeavers*, mas depois de um certo tempo, devido aos esforços do usuário *thedoctor45*, o port não apenas se tornou utilizável para outras aplicações como se tornou possível trocar seu motor pelos das versões mais recentes do *CrossOver*, os quais eram abertos devido ao fato de serem baseados no *Wine*.

Esse wrapper e suas variações começaram a se tornar amplamente utilizados. O original era conhecido como *Chromium Wrapper*, enquanto suas variações foram nomeadas *CX Wrappers*, ou *CrossOver Wrappers*. Isso deu origem ao primeiro tipo de wrapper que poderia ser modificado e adaptado com mais facilidade pela comunidade.

Um certo tempo depois, em 30 de junho de 2009, o usuário *Doh123* introduziu na *The Game Porting Team* o *OSXDosBoxWrapper*, um wrapper voltado para portar aplicações em *DOS* para o *Mac OS X*, marcando o primeiro wrapper criado por um usuário em vez de uma empresa. Depois da popularidade do seu wrapper, em 5 de julho do mesmo ano, *Doh123* criou então o *WineWrapper*, com o objetivo de portar aplicações de *Windows* para *Mac* usando motores *Wine*. Este programa foi rebatizado pouco depois para *Wineskin*.

Figura 3: Antigo logo oficial do Wineskin

Cerca de dois meses depois, a *The Game Porting Team* saiu do ar duas vezes por falta de pagamento devido à falta de doações. Isso acabou a levando a migrar para uma hospedagem mais barata, fazendo surgir assim a *Porting Team*, em um novo domínio (http://portingteam.com/) que começou a vigorar no dia 24 de setembro de 2009.

O *Wineskin* usava o *XQuartz* para se renderizar, e se popularizou com uma velocidade abrupta, mas os motores do *CrossOver* ainda continuavam a ser melhores que os motores *Wine* para jogos e aplicações, além de não dependerem do *XQuartz*, o que mantinha os *CX Wrappers* em alta, por mais que não houvessem versões dos mesmos com os motores mais recentes. Foi aí que, aproximadamente em Outubro do mesmo ano, *Doh123* criou o *CXSkin*.

Utilizando o mesmo *X11* contido dentro dos *CX Wrappers* e motores *CrossOver* recompilados em um wrapper Wineskin, foi possível criar um wrapper facilmente editável que usava motores *CrossOver*. Este novo tipo de wrapper não era muito diferente do *Wineskin*, mas o fato de utilizar motores *CrossOver* o tornava um substituto para os *CX Wrappers*, os quais inclusive já não funcionavam no sistema mais recente da época, o *Mac OS X 10.6* (*Snow Leopard*). A intenção de *Doh123* era apenas descobrir como o *X server* funcionava, mas as pessoas começaram a usá-lo como um *CrossOver* gratuito, o que o levou a descontinuar o *CXSkin*.

Então, em novembro de 2009, o usuário *Devilhunter* usou o código do *CXSkin* para criar o *CXZ*, um wrapper que possuía um único diferencial em relação

ao seu antecessor: seu motor era externo. Sendo colocado em um diretório do sistema em vez de dentro do wrapper, um mesmo motor poderia ser usado com múltiplos ports, o que trazia uma economia de espaço. No decorrer de seu desenvolvimento, assim como o *Wineskin*, os wrappers *CXZ* passaram a carregar dentro de si um ou mais programas que tinham como objetivo facilitar ainda mais a modificação dos wrappers.

Figura 4: Logo do CXZ na comunidade Porting Team

Em um momento desconhecido, mas que estima-se ter ocorrido no início de 2010, a companhia *The Bordeaux Technology Group* lançou no mercado a aplicação paga *Bordeaux for Mac OS X*. Era basicamente uma janela como a do Windows Explorer, na qual apareciam as aplicações instaladas pelo programa, o qual tinha suporte a apenas um motor e um único local para instalação. Por ser bem similar ao *WineBottler*, uma solução que já existia, e ainda ser pago, não recebeu destaque algum na comunidade.

Figura 5: Logo oficial do Bordeaux for Linux / Mac OS X

Depois de aproximadamente um ano de alterações e aprimoramentos no *CXZ*, *Devilhunter* lançou o *CXEx* no final de 2010, uma versão aprimorada do mesmo que possuía um gerenciador de motores, permitindo que eles fossem baixados por meio deste gerenciador em vez de serem baixados manualmente pelo usuário. Além do gerenciador, o *CXEx* também introduziu a sua própria janela para alteração do wrapper, o que antes era um ponto negativo do *CXZ* já que o *Wineskin* havia introduzido uma única janela para fazer alterações há bastante tempo, enquanto os wrappers *CXZ* possuíam vários programas, cada um com pequenas funcionalidades.

Figura 6: Logo oficial do CXEx

Pouco antes do lançamento do *CXEx*, o usuário *Drakulix* introduziu na comunidade o *CiderX*. Basicamente se tratava de um port *CXZ*, mas com modificações que lhe permitiam usar motores Cider além dos motores *CXZ* já existentes. Isso abrangia muito seu número de apps suportados, mas ele acabou se tornando dependente do *CXEx* para o download de motores depois que este surgiu, o que fazia com que se tornasse apenas uma expansão dele.

Figura 7: Logo oficial do CiderX

Numa tentativa de se recuperar, a *Bordeaux* lançou em 17 de março de 2011 sua própria versão do *Wineskin*, a qual chamou de *Wineskin Pro*. O produto era essencialmente o mesmo, no entanto eles ofereciam também suporte para auxiliar na instalação de aplicações, e este seria o seu diferencial, ou ao menos era o que deveria ser. Devido ao péssimo suporte oferecido e à mesma aplicação também estar disponível gratuitamente na internet, o Wineskin Pro durou apenas um pouco mais de dois anos, sofrendo inclusive uma redução de preço, até que a companhia o descontinuou como se nunca tivesse acontecido. Ele nunca chegou a ter destaque na *Porting Team*.

Figura 8: Logo oficial do Wineskin Pro

Os wrappers *CXSkin* (agora também conhecidos como *CXS*) e *CXZ* pararam de funcionar a partir da versão 10.7 do sistema da Apple. Enquanto isso, o *CXEx* perdurou até a atualização dos sistemas *Lion* e *Mountain Lion* para OS X 10.7.5 e OS X 10.8.2 em 19 de setembro de 2012, quando então passou a apresentar mal funcionamento. Apesar disso, o mesmo já não tinha o suporte de seu criador desde 8 de fevereiro de 2011, depois que a versão 2.0 do Wineskin foi lançada. Por consequência, o *CiderX* também se tornou obsoleto, não por parar de funcionar, mas por depender do *CXEx*, o qual já não era mais suportado.

Por via de curiosidade, pesquisei sobre os motores utilizados pelos wrappers CXS, CXZ e CXEx. Descobri que se tratavam de apenas 14 motores ao todo, todos baseados em motores do CrossOver, os quais por sua vez são derivados de motores do Wine com diversas modificações aplicadas.

Motor	Versão do Wine Original	Versão do CrossOver Correspondente
Platinum	Wine 1.4	CrossOver 11.0
Carbon	Wine 1.3.9	CrossOver Games 10.0.0
Chrome	Wine 1.3.9	CrossOver Pro 10.0.0
Ebony	Wine 1.2.1	CrossOver Games 9.2.0
Ivory	Wine 1.2.1	CrossOver Pro 9.2.0
Aquamarine	Wine 1.1.42	CrossOver Games 9.0
Jade	Wine 1.1.35	CrossOver Pro 9.0
Black Diamond	Wine 1.1.25	CrossOver Games 8.1.3
Diamond	Wine 1.1.25	CrossOver Games 8
Sapphire	Wine 1.1.18	CrossOver Pro 8.0
Ruby	Wine 1.1.12	CrossOver Games 7.2
Topaz	Wine 1.1.4	CrossOver Pro 7.1.1
Amethyst	Wine 1.1.0	CrossOver Games 7.1.1
Emerald	Wine 0.9.60	CrossOver Pro 7.1

Tabela 1: Motores CXZ/CXEx e seus correspondentes

Um detalhe interessante sobre esses motores é que o mais recente deles (Platinum) foi criado pelos usuários *thedoc* (antes conhecido como *thedoctor45*) e *waves* depois que os projetos CXS, CXZ e CXEx já haviam sido encerrados, o que faz com que ele só possa ser encontrado em ports criados por eles para usuários de sistemas antigos e em derivados destes ports.

Devido ao fato de continuarem recebendo atualizações por parte de seu desenvolvedor, os wrappers Wineskin foram os únicos a sobreviverem a todas essas atualizações. Além disso, em determinado momento, *Doh123* passou a usar uma versão modificada do *XQuartz* dentro de cada wrapper chamada *WineskinX11*,

o que levou à total independência dos wrappers Wineskin de qualquer software externo. Atualmente, os motores CrossOver também estão incluídos na lista de motores do Wineskin.

Figura 9: Atual logo oficial do Wineskin

Em resumo, esta é a avaliação dos diferentes tipos de wrappers que podem ser encontrados na internet, seus lançamentos, seus preços e até qual SO são compatíveis:

	Data de lançamento	Compatível	Preço
Cider	Outubro de 2006	**Mac OS X 10.4 - macOS 10.14**	Empresarial
CrossOver Wrapper	15 de Setembro de 2008	Mac OS X 10.4 - Mac OS X 10.5	Gratuito
OSXDosBoxWrapper	30 de Junho de 2009	Mac OS X 10.4 - OS X 10.7.4 / 10.8.1*	Gratuito
WineWrapper / Wineskin	5 de Julho de 2009	Mac OS X 10.4 - OS X 10.7.4 / 10.8.1*	Gratuito
CXSkin / CXS	Outubro de 2009*	Mac OS X 10.4 - Mac OS X 10.6*	Gratuito
CXZ	Novembro de 2009	Mac OS X 10.4 - Mac OS X 10.6*	Gratuito
CiderX	9 de Setembro de 2010*	Mac OS X 10.? - OS X 10.7.4 / 10.8.1	Gratuito
CXEx	Novembro de 2010*	Mac OS X 10.4 - OS X 10.7.4 / 10.8.1	Gratuito
Wineskin 2.0 - 2.5.4	8 de Fevereiro de 2011*	Mac OS X 10.5 - OS X 10.7.4 / 10.8.1	Gratuito
Wineskin Pro	17 de Março de 2011	Mac OS X 10.5 - OS X 10.7.4 / 10.8.1	US$ 29,95 / US$ 4,99
Wineskin 2.5.5+	18 de Abril de 2012	Mac OS X 10.6 - OS X 10.7.4 / 10.8.1	Gratuito
Wineskin 2.5.8+	21 de Setembro de 2012	Mac OS X 10.6 - OS X 10.10	Gratuito
Wineskin 2.6.1+	4 de Outubro de 2015	**Mac OS X 10.6 - macOS 10.14**	Gratuito

Tabela 2: Os diferentes tipos de port

** Dados indicados com um asterisco (*) não foram testados e podem não ser precisos.*

5. FUNCIONAMENTO

Sem dar explicações detalhadas, mesmo um usuário técnico pode ficar cético a como o Wine e seus derivados funcionam. Como isso se trata de um tópico essencial para explicar o funcionamento do próprio Porting Kit, será explicado como o Wine funciona, e porque de todos os derivados do Wine, o Wineskin foi o escolhido.

As opções existentes até o momento em que o projeto não havia sido iniciado eram as que seguem: CrossOver, PlayOnMac, Cider, WineBottler, CXZ, CXEx e Wineskin. Como todas possuem suas vantagens e desvantagens, a decisão foi tomada por um processo de eliminação.

Primeiramente foi considerado o custo. De todas as opções citadas, o CrossOver é a única solução que cobra pelos seus serviços. Dado isso, ele não foi considerado uma opção.

Em seguida pensou-se na questão da aparência nativa. O quão similar a aplicativos do próprio sistema ficavam os ports de cada um? Aqueles que mais se distanciaram eram os instalados pelo PlayOnMac e pelo WineBottler, os quais não foram utilizados por isso.

O terceiro quesito foi a flexibidade, ou em outras palavras, o quão variados poderiam ser os ports criados por cada um dos tipos de wrapper restantes. Cider, por se tratar de uma solução vendida para empresas, era distribuído de forma a ser moldado para aplicações específicas, além de não possuir ferramentas que permitissem a instalação de novos recursos. Devido a isso, seu uso foi descartado.

Isso deixou apenas como possibilidades o CXZ, o CXEx e o Wineskin. Tanto o CXZ quanto o CXEx eram baseados no Wineskin, com a diferença de que ambos deixavam seus motores do lado de fora dos ports, para evitar replicação. A consequência disso é que um usuário mal informado poderia apagar o motor ao realizar uma limpeza do sistema, sem saber que a aplicação depende dele para funcionar. Além disso, os utilitários do CXZ e do CXEx eram muito menos práticos do que o utilitário presente no Wineskin, caso o usuário precisasse usá-lo. Isso levou por fim, à decisão de usar o Wineskin.

5.1 - WINESKIN

O funcionamento do Wineskin em si é derivado diretamente do Wine.

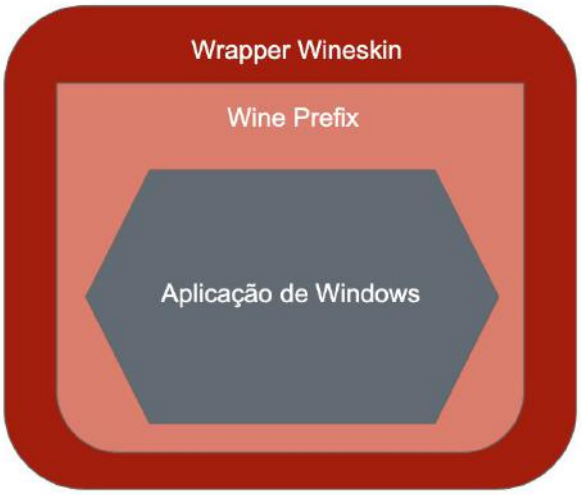

Figura 10: Arquitetura do Wineskin

Como pode se pode observar na figura 10, o Wineskin pode ser descrito por meio de duas camadas ao redor da aplicação de Windows. O Wine Prefix nada mais é do que uma instalação do Wine, contida dentro de um Wrapper Wineskin, o qual mantém tudo encapsulado na forma de uma aplicação do sistema macOS.

Desta forma, se torna possível possuir várias instalações do Wine na mesma máquina, e ainda permite que todas elas possuam uma aparência nativa.

5.2 - PORTING KIT

Como mencionado anteriormente, a ideia base do Porting Kit é permitir que um port possa ser replicado por qualquer pessoa. Para isso, é necessário que um usuário mais avançado inicialmente desenvolva o port de aplicação para o macOS, o qual pode ser feito com a ajuda do Porting Kit se o mesmo desejar, ou não; o único pré-requisito para a compatibilidade, é que esse port seja feito com Wineskin.

Uma vez com o port pronto, o usuário só precisa usar o Porting Kit para criar um arquivo WSI baseado neste port, o qual possuirá em si o conhecimento de como recriar aquele port.

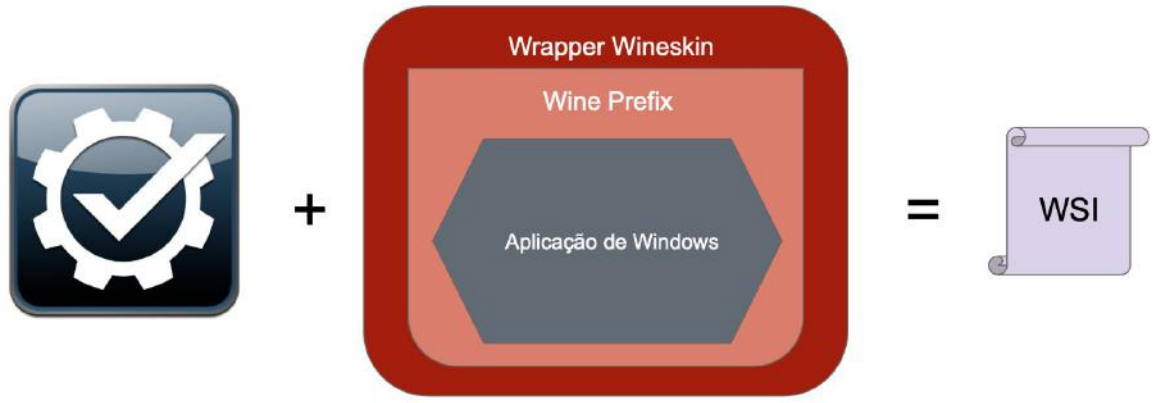

Figura 11: Fluxograma do usuário avançado no Porting Kit

Este usuário poderá então enviar esse arquivo WSI para um usuário leigo, que só precisará ter o Porting Kit instalado em sua máquina, e um instalador original daquela mesma aplicação de Windows, para que ele possa criar o port em sua máquina.

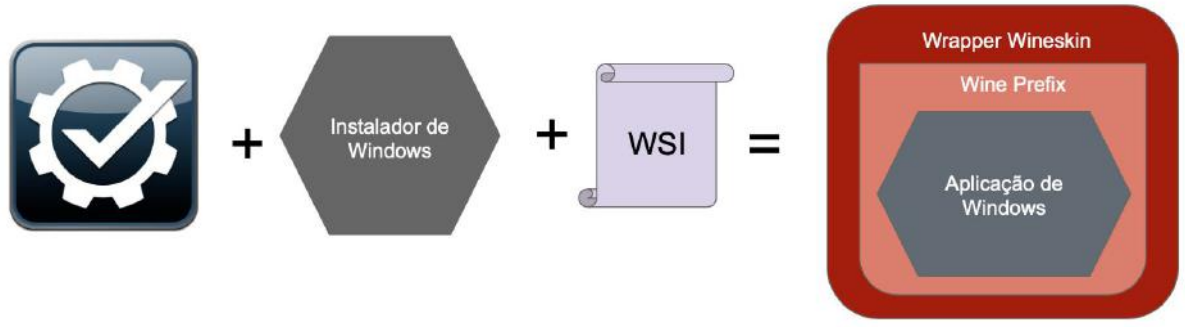

Figura 12: Fluxograma do usuário leigo no Porting Kit

Isso torna o sistema adequado para ambos os usuários, avançado e leigo, provendo o melhor resultado com o menor custo.

6. EVOLUÇÃO

Em agosto de 2011, comecei a desenvolver o projeto pelo nome de ***Appleverse Wizard***. O nome foi pensado unindo as palavras *Apple* e *Universe*, pensando no universo da Apple como os computadores Mac, e *Wizard* pois ele funcionava como uma aplicação para facilitar processos. O ícone utilizado neste projeto está apresentado na figura 13.

Figura 13: Ícone utilizado no Appleverse Wizard

A linguagem utilizada foi *Java*, e o programa não estava servindo ao seu propósito. Ele podia basicamente alterar configurações básicas de um wrapper (que é o pacote onde um programa de Windows é armazenado para que funcione no Mac OS X), criar um drive D dentro do wrapper e converter uma imagem PNG quadrada ou um arquivo ICO para ICNS (formato de ícone do Mac OS X, que pode ser usado como ícone do port). A janela principal do aplicativo é mostrada na figura 14.

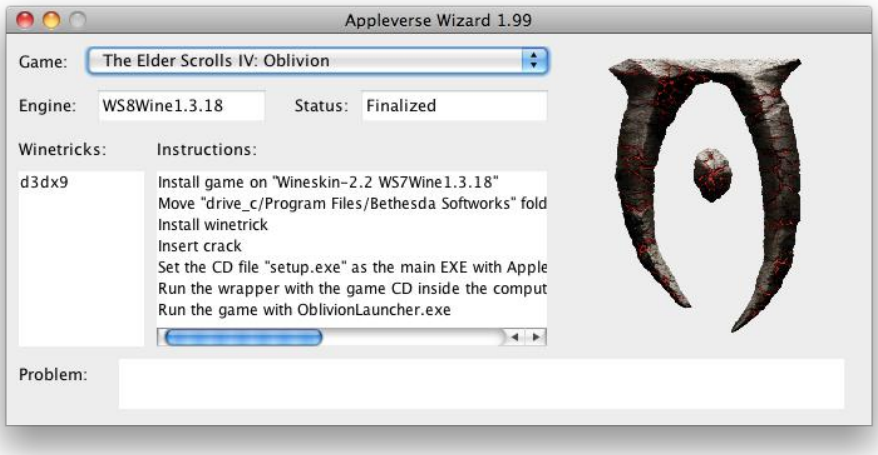

Figura 14: Janela do Appleverse Wizard em sua versão final

Depois de alguns meses, vi que prosseguir o desenvolvimento em Java apenas o atrasaria devido ao excesso de código necessário para fazer tarefas simples, e à falta de recursos específicos do macOS que seriam necessários. Sendo assim, decidi que deveria mudar de linguagem assim que fosse possível.

Quando se tornou viável, comprei um livro de Objective-C dos Estados Unidos: "*Beginning Mac Programming - Develop with Objective-C and Cocoa"* de Tim Isted, e por meio dele recomecei o projeto do zero chamando-o de **Wigidrasil**. O ícone utilizado no novo projeto pode ser visto na figura 15.

Figura 15: Ícone utilizado no Wigidrasil

O nome Wigidrasil vem da abreviação em inglês W.I.G.I., *Wineskin Instructions for Games Installation*, o que pode ser traduzido para o português como *Instruções do Wineskin para Instalação de Jogos,* combinada à palavra *Yggdrasil*, uma árvore da mitologia nórdica que se dizia conectar os diferentes mundos, que era um dos grandes propósitos do Wigidrasil.

Apesar de seu nome representar isso, o Wigidrasil perdeu o foco de ser um sistema de tutoriais e passou a ser um editor de ports. No começo ele só suportava o Wineskin, mas conforme o tempo passou fui dando suporte às suas variações também.

Nesta fase, o desenvolvimento progrediu bastante, e depois de muito progresso, em fevereiro de 2013, decidi que o programa passaria de Beta para Estável. Consequentemente, achei que o nome deveria mudar, já que a idéia de um sistema de instruções para Wineskin estava morta a algum tempo.

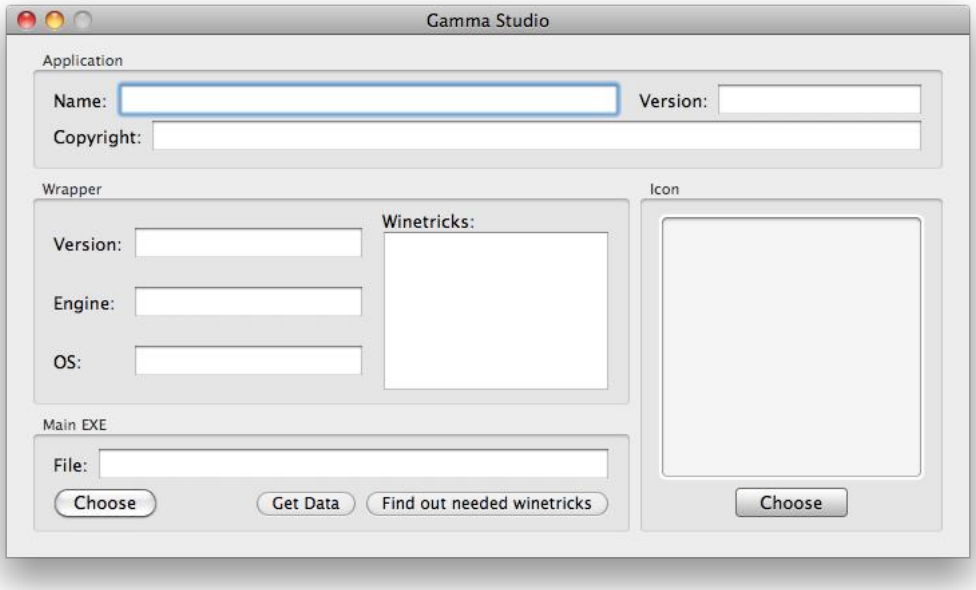

Figura 16: Última janela do Wigidrasil / Primeira janela do Gamma Studio

A figura 16 mostra como estava a janela principal do Wigidrasil quando o projeto mudou seu nome. A ideia do projeto neste momento era que ele fosse capaz de editar os wrappers dos mais variados tipos, mas que também fosse capaz de gerenciar todo eles, como o programa da empresa *Steam* faz com seus jogos. Então, surgiu o nome ***Gamma Studio***. *Gamma* advém da abreviação de **Gam**e **Ma**nager, enquanto o *Studio* é devido ao fato de ser um programa de edição, e a palavra remeter a isso. A figura 17 mostra a evolução do ícone do Gamma Studio.

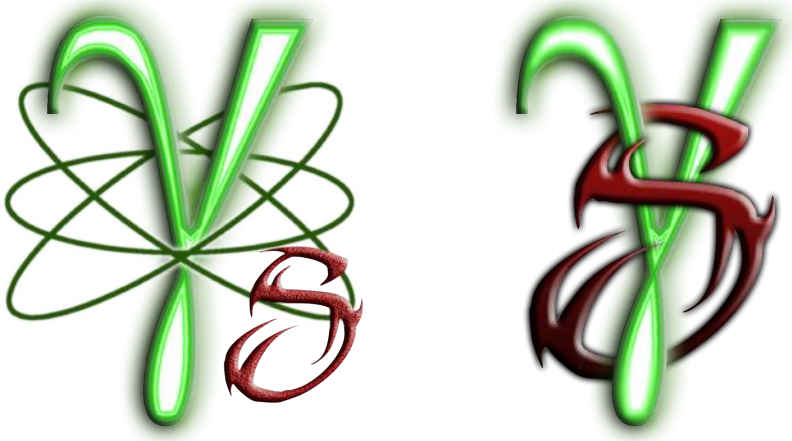

Figura 17: Evolução do ícone do Gamma Studio

Nesse mesmo tempo, comecei a desenvolver outro programa que seria distribuído junto ao Gamma Studio como o seu gerenciador de ports, o **Gamma Board**, o qual o ícone provisório pode ser visto na figura 18. No entanto, o mesmo foi deixado de lado quando alguns questionamentos me vieram à mente: Por que o Gamma Studio existe? Qual o seu grande propósito? Por que eu o uso no lugar dos utilitários originais dos wrappers?

Figura 18: Ícone utilizado no Gamma Board

Foi quando observei que a capacidade de editar os diferentes tipos de wrappers acidentalmente transformou o Gamma Studio em uma interface universal para a edição dos mesmos. Isso, combinado ao fato do Gamma Studio possuir recursos exclusivos que facilitam a edição, o torna único e necessário. Combinando isso à sua praticidade, o Gamma Studio se tornou a ferramenta perfeita para o trabalho.

Com isso, continuei no desenvolvimento do projeto focado principalmente na edição dos wrappers, e também dei ênfase à tradução da interface. Com a ajuda de dois usuários que conheci em comunidades referentes aos ports, DankoB e Evilence, consegui traduzir o Gamma Studio para Russo e Francês, e com a ajuda de meu pai e de uma de minhas tias, consegui traduzi-lo também para Espanhol e Italiano. Combinando isso às minhas próprias traduções para Inglês e Português, o Gamma Studio totalizou 6 idiomas, contra apenas o inglês usado pelos outros aplicativos.

O grande objetivo do Gamma Studio se tornou editar wrappers de todos os 5 tipos variados do Wineskin, que apesar de terem a mesma origem, se tornaram muito distintos em suas interfaces.

Os recursos adicionais do Gamma Studio tinham como foco principal a praticidade e a economia de tempo, os quais serão citados em mais detalhes no capítulo 2, porém é possível dar um exemplo: para se extrair um ícone de uma aplicação de Windows da maneira ideal para usá-lo em seu port, sem a ajuda do Gamma Studio, é necessário utilizar dois programas que funcionam via terminal, e em seguida utilizar um programa de terceiros, o que pode ter falha humana e com isso perda de qualidade. Com o Gamma Studio, é necessário apenas arrastar a aplicação até a janela do mesmo.

Por fim, depois de muito tempo de desenvolvimento, foi iniciada uma parceria com o dono do site paulthetall.com, o qual é um dos maiores nomes em porting dos tempos atuais, para que o projeto do programa Gamma Board fosse retomado, recebendo de Paul Bertelink não apenas suporte financeiro, mas também seus wrappers, que seriam usados futuramente na criação dos arquivos WSI que serão explicados em 6.1.12 e 6.2.6.

Paul financiaria os servidores, nos quais foram hospedados os arquivos WSI gerados pelo Gamma Studio, enquanto o Gamma Board baixaria o arquivo WSI de uma aplicação específica toda vez que fosse requisitado. Isso tornaria o Gamma Board um programa para o usuário comum de Mac, e o Gamma Studio seria uma aplicação para desenvolvedores que quisessem seu programa de Windows funcionando no Mac.

Com isso, em uma conversa com Paul, foi decidido que o nome do Gamma Board passaria a ser *Porting Kit*, e posteriormente foi decidido que o nome do Gamma Studio deveria ser *Porting Center*. Com a ajuda do designer Stuart Ludwig, amigo de Paul, foram desenhados os ícones finais para ambos os programas, conforme apresentados na figura 19.

Figura 19: Novo ícones do Porting Center e do Porting Kit respectivamente

Apesar de tudo isso, coisas inesperadas surgiram no caminho. Durante a programação do recurso de instalação de *winetricks* (explicado no capítulo 6.1.11) foi descoberto durante o download de *ports* para testes que havia um sexto tipo de *wrapper*: uma versão mais antiga do CXEx, a qual parecia um híbrido do novo CXZ e do CXEx, batizado de **antigo CXEx**. Suas similaridades com esses dois tipos de *wrapper* tornaram a sua adição ao Porting Center muito mais simples, mas inspiraram à uma nova possibilidade: adicionar suporte ao CrossOver *read-only*, ou seja, apenas para a geração de arquivos WSIs usando os arquivos TIE do mesmo (isso foi implementado em agosto de 2015).

Mesmo com tudo isso, as descobertas nunca tinham fim. No início de Junho de 2015 foi descoberto que o tipo antes batizado de **antigo CXZ** na realidade era um **Wineskin Beta**, e que havia mais um outro tipo antes do CXZ, o **CXSkin** (combinação dos nomes CrossOver e Wineskin). O CXSkin teria sido criado utilizando a estrutura Wineskin com motor CrossOver Games usando o motor externamente.

O CXZ teria sido criado como um sucessor do CXSkin, assim como o CXEx se tornou o sucessor do CXZ. No fim, essa *linhagem* de wrappers acabou se extinguindo, o que tornou seus wrappers inúteis para quem utiliza sistemas mais modernos, exceto para os usuários do Porting Center, pois esses seriam capazes de convertê-los em ports Wineskin.

Em novembro do mesmo ano, mais um tipo já *extinto* de wrapper foi encontrado: o **wrapper CX**, o qual teria sido desenvolvido originalmente pela própria

Codeweavers como um port para a aplicação Chromium. A descoberta desse wrapper levou a uma pesquisa sobre a história dos ports e sua evolução (que desmentiu várias das minhas suposições anteriores), a qual pode ser encontrada no capítulo 4.

Os programas prosperaram, e isso deu início ao que batizamos de **PROJETO PORTING**, que trata da junção do uso dos programas Porting Center e Porting Kit para se alcançar o objetivo deste projeto.

Apesar dos avanços, o surgimento do El Capitan em 2015 acarretava mudanças no sistema de permissões do Mac OS X, e foram necessárias mudanças no Wineskin e no próprio Porting Kit para permitir que novos wrappers funcionassem nesse sistema. Com um pouco mais de trabalho também foi possível permitir ao Porting Kit recuperar wrappers criados para versões mais antigas do sistema.

No final de 2015, o Porting Kit e seu servidor tiveram uma grande atualização que abriu novas possibilidades para o Porting Project, como um sistema de busca avançada online e uma análise mais profunda dos ports disponíveis. Mais informações serão explicadas no capítulo 6.2.

Em maio de 2017, os recursos do Porting Center foram movidos para o Porting Kit, tornando-os uma única aplicação. Isso foi feito para resolver as instabilidades que surgiram com o tempo no Porting Center, e ainda promover a integração entre o ambiente de desenvolvedor e o ambiente do usuário comum.

E então surgiram outras perguntas: Para que tudo isso? Qual a grande utilidade do Porting Kit? E dessa vez pude observar que combinando os ports ao Projeto Porting, tínhamos a integração ideal entre o sistema Mac OS X e os programas de Windows.

O encapsulamento das aplicações instaladas pelo Porting Kit e o fato de seus ports não dependerem dele para funcionar tornam ele uma melhor opção do que os concorrentes CrossOver, PlayOnMac e WineBottler para todos os tipos de usuário, com exceção daqueles que precisam de recursos disponíveis apenas nos motores do CrossOver mais recentes, como certas funções do DirectX 11. Apesar disso, o fato de ser uma aplicação gratuita o torna uma opção mais viável para instituições que precisem utilizar seus serviços em diversas máquinas por exemplo, diferentemente do CrossOver, o qual proíbe em sua política que uma mesma

licença seja usada por mais de uma pessoa, exigindo uma licença por máquina, as quais variam de US$ 29.95 a US$ 499.95.

A curto prazo, essa era uma solução para os problemas dos usuários de macOS de todo o mundo que precisam (ou desejam) usar certas aplicações que normalmente não estão presentes neste ambiente. A longo prazo, pode induzir à criação de outros ‘portadores’ de um sistema para outro, e resultar futuramente no fim da rixa entre os sistemas operacionais, fazendo com que os usuários escolham um sistema por sua praticidade, e não seus programas ou jogos, pois ambos terão todas as opções de software.

A intenção é que o produto final seja mais fácil de contribuir que o PlayOnMac, mais versátil que o CrossOver, mais portátil que o WineBottler, e mais simples e acessível do que todos.

No capítulo 2, o funcionamento do Porting Center será explicado em detalhes, assim como o processo de desenvolvimento dos seus recursos, antes de serem movidos para o Porting Kit. No capítulo 3, a aplicação Porting Kit será detalhada, assim como seus recursos próprios, para melhor entendimento sobre o mesmo.

O capítulo 4 descreve a evolução dos diferentes tipos de ports em ordem cronológica, mostrando como foi seu surgimento e desenvolvimento, e explicitando por que o Wineskin se manteve como o tipo de wrapper criado pelo Porting Kit.

Algumas das diferentes aplicações para as quais o Porting Kit pode ser empregado podem ser encontradas no capítulo 5.

6.1 - PORTING CENTER

Um wrapper consiste basicamente em uma miniatura de um computador com o sistema operacional Windows, que possui apenas os arquivos necessários para a execução de seus aplicativos. Ele conta com seus próprios drives e seu próprio registro, podendo assim suportar qualquer gama de aplicativos.

Dentro de todo wrapper (que também pode ser reconhecido pelo sistema como uma pasta) existe a pasta *Contents*, dentro da qual estão todos os arquivos do programa. Esta pasta costuma se dividir em *MacOS*, *Resources* e *Info.plist*,

dentre as quais: *MacOS* é a pasta do programa responsável pelo port; *Resources* é a pasta que contém, dentre outras coisas, os drives do port, seu ícone e seu registro; e Info.plist é um documento que contém todas as informações necessárias para que o port seja executado, como qual o EXE de Windows que deve ser executado quando o port for aberto.

Alguns wrappers contém pastas e arquivos extras para realizar sua função, como o Wineskin, que conta com a pasta *Frameworks* dentro da pasta *Contents*, a qual possui as bibliotecas de redirecionamento, que fazem as instruções do port serem passadas para o macOS, permitindo coisas simples como editar um texto ou até mais complexas como renderização de objetos 3D.

No decorrer do seu desenvolvimento, diversas funcionalidades foram criadas e inseridas no Porting Center para suprimir as necessidades dos usuários do sistema macOS que utilizam wrappers, tornando o processo de trabalhar com eles mais rápido e mais simples. Algumas delas eram simples derivações de funcionalidades de outros utilitários, enquanto outras eram exclusivas do Porting Center, dando-lhe um diferencial.

Conforme o tempo se passou novos recursos foram se tornando necessários, e com isso mais dados precisavam ser enviados pelo Porting Center para o servidor. Eventualmente, como esses dados não haviam sido previstos originalmente, a aplicação se tornou muito instável, ao ponto de se tornar insustentável. Para resolver o problema, os recursos do Porting Center foram totalmente reescritos no Porting Kit em maio de 2017.

Nos próximos subcapítulos serão apresentadas todas as funcionalidades do Porting Center, bem como a forma que como foram desenvolvidas e implementadas, e de que maneira foram adaptadas para o Porting Kit. Uma lista incompleta das mudanças que ocorreram no Porting Center no decorrer de sua evolução pode ser encontrada no anexo 1.

6.1.1 - Janela Principal

Como em qualquer outro programa, a primeira coisa que se tem ao iniciar o Porting Center é a janela principal. A distribuição de recursos pela tela pode parecer

simples, mas até chegar a esse ponto houveram diversas mudanças, o que torna a janela do Porting Center um recurso exclusivo e essencial.

Desde o Appleverse Wizard, passando pelo Wigidrasil e pelo Gamma Studio até o Porting Center, diversas janelas foram desenvolvidas para o programa, e muitas mudanças ocorreram com o decorrer do tempo.

Como pode se observar na figura 14, o foco do Appleverse Wizard eram as instruções para a instalação de certos aplicativos (especialmente jogos), e isso o tornava muito limitado, além de deixar escondidas algumas funções importantes. Quando se tornou Wigidrasil, figura 16, pode-se notar que as funções realmente úteis ganharam destaque.

No Wigidrasil, os recursos que hoje são disponibilizados nas abas inferiores eram encontrados em janelas separadas. No início isso parecia poupar espaço na janela principal, mas acabou tornando o programa cheio de janelas, o que era uma característica ruim que gerava muita confusão durante o uso de janelas simultâneas.

Considerando que o antigo *'Main EXE'* era pequeno demais para fazer parte do conjunto de abas inferiores, e já tendo em mente que futuramente se tornaria a aba *'EXEs'*, o 'Main EXE' foi inserido ao lado da aba *'Port'*, criando a dupla *'EXEs'* e *'Port'* posteriormente, como mostrado na figura 20.

Com isso, todas as outras janelas se tornaram abas na área inferior do programa, sendo elas *'Avançado'*, *'Drives'*, *'Barra de Menu'* e *'Tela' (com 'Extensões' vindo apenas no futuro)*. Por um tempo, cogitou-se levar a aba *'Drives'* para a barra de abas do centro, no entanto, devido aos 5 botões necessários na aba *'Drives'*, assim como a possibilidade de haverem caminhos (*paths*) longos para os drives, isso foi descartado.

Por fim, o Gamma Studio alcançou sua estrutura final de janela, conforme apresentado na figura 20.

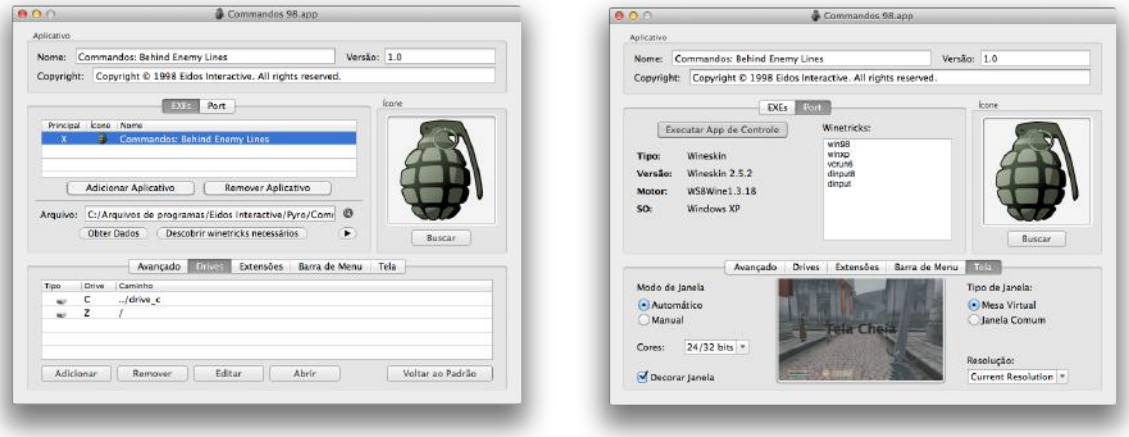

Figura 20: Janela do Gamma Studio abrindo um port

As abas *'EXEs'*, *'Port'* e *'Tela'* foram as que mais sofreram alterações desde que o programa foi criado, tendo ganhado mais recursos com o tempo, como a caixa de seleção de Cores em *'Tela'*, o SO (Sistema Operacional) em *'Port'* e a barra de gerenciamento de Custom EXEs (executáveis secundários num mesmo port) em *'EXEs'*.

Vale lembrar que outras coisas foram inseridas em sua interface depois de mais algum tempo: o botão de instalação de winetricks, novos recursos na aba de tela, entre outros; mas tudo isso será explicado de forma mais aprofundada nos próximos subcapítulos.

Quando o Porting Center foi integrado com o Porting Kit, a janela do Porting Center foi mantida, mas modificada. Como todo Port no servidor é Wineskin, o SO só pode ser mudado propriamente por winetricks, a versão do Wineskin era irrelevante, o motor e os winetricks seriam gerenciados em outra tela e o Wineskin poderia ser chamado de outro lugar, dados esses que serão explicados mais adiante, essas partes foram removidas da janela, promovendo assim um layout diferente, como visto na figura 21.

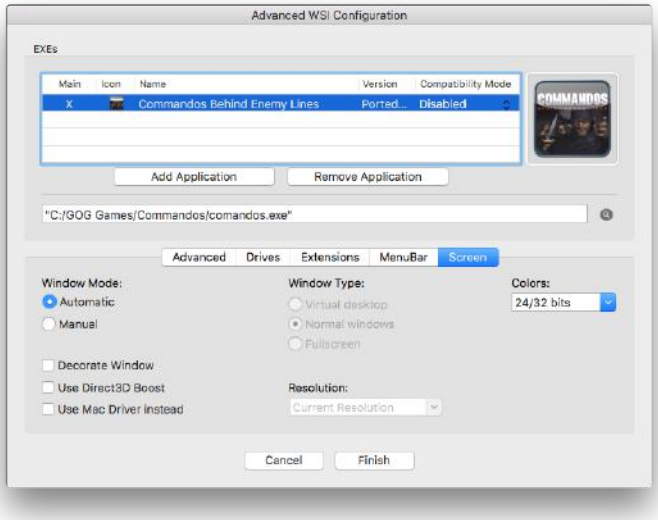

Figura 21: Janela de exportação avançada de WSI no Porting Kit

6.1.2 - Obtenção de ícone

Nos dias de hoje, independente de qual seja o sistema operacional, todos os tipos de aplicativo são identificados pelo seu ícone. No *Windows* esse ícone pode estar atribuído a um aplicativo (.exe) ou pode ser salvo no formato específico para o mesmo (.ico), porém, no *Mac OS X* os ícones de aplicativos se encontram sempre no formato ICNS (.icns).

Uma das dificuldades enfrentadas pelos usuários ao se instalar uma aplicação de *Windows* no *Mac* por meio de um wrapper é a ausência de um ícone. Normalmente, um ícone para uma aplicação pode ser encontrado rapidamente em uma pesquisa no *Google*, mas além de demandar tempo e trabalho com a pesquisa, o ícone encontrado pode não ser satisfatório, e ainda precisaria ser convertido para o formato ICNS, exigindo o uso de mais um programa de terceiros, como o *img2icns*, podendo não ter resultados satisfatórios, já que o mesmo proporcionava um clareamento na imagem, como pode-se ver na figura 22.

Figura 22: Ícone de “The Elder Scrolls IV: Oblivion” antes e depois da conversão pelo img2icns.

O Porting Center fornece um leque de extensões (diferentemente dos outros utilitários, que suportam corretamente apenas a extensão ICNS, e às vezes imagens quadradas) pois ele é capaz de realizar sua conversão para o formato aceito pelo macOS. Alguns dos formatos que possuem suporte são:

- Imagens suportadas pelo programa “Pré-Visualização”, cujo ícone está representado na figura 23 (Exemplo.: JPG, BMP, PNG, GIF, TIFF, PSD. etc);

Figura 23: Ícone do programa Pré-Visualização

- Vídeos suportados pelo programa “QuickTime”, onde o ícone é a imagem de pré-visualização que o próprio sistema usa como ícone, como mostrado na figura 24;

Figura 24: Exemplo do resultado da extração de ícone de um arquivo de vídeo

- Qualquer coisa que possua uma pré-visualização no QuickLook, como pastas e arquivos diversos, como mostrado na figura 25;

Figura 25: Ícones obtidos a partir de uma pasta comum, do diretório raiz (HD) e de um diretório Home

- Imagens vetorizadas SVG (figura 26), que são rasterizadas pelo *rsvg-converter* e por isso possuem uma excelente qualidade;

Figura 26: Exemplo de uma imagem SVG após a rasterização

- Aplicativos EXE do Windows, com ICO extraído pelo *wrestool*, o qual são uma ótima opção no caso de programas recentes, como representado na figura 27;

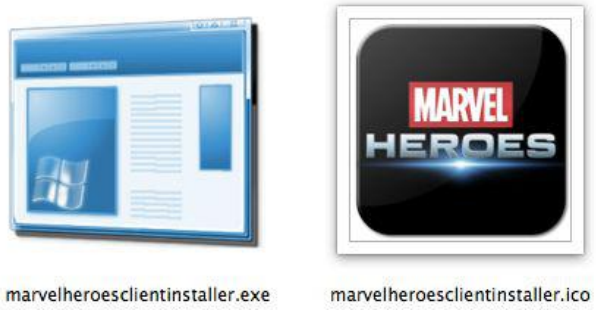

Figura 27: Exemplo de extração de ICO do EXE instalador do jogo “Marvel Heroes"

- Ícones ICO de Windows, com imagem extraída pelo *icotool* (figura 28), já que as vezes uma nova extração é necessária para que o ícone seja visualizado.

Figura 28: Exemplo de extração de PNG do ICO extraído do executável do jogo "Mass Effect"

Os métodos usados para obter as imagem representadas nas figuras 26, 27 e 28 serão explicados em mais detalhes por possuírem um algoritmo envolvido (EXE e ICO) ou por terem demandado uma pesquisa bem maior para que uma forma de extração fosse executada (SVG). Também será explicado um recurso de aprimoramento de imagem bem versátil que aprimora a obtenção de ícone.

6.1.2.1 - Extração de ícone de .EXE

Um ícone inadequado para um port não ocorreria se o ícone do programa original estivesse sendo usado (isto é, as aplicações mais antigas possuem uma

resolução muito baixa, mas a solução para esse problema será apresentada em 6.1.2.3). Isso pode ser superado usufruindo de mais um recursos do *Porting Center*, que é a extração de ícone de EXE.

Como houve diversos testes com diferentes executáveis de *Windows*, foi possível observar que diferentes executáveis desse ambiente podem armazenar seus ícones de maneira diferente, o que exige a criação de um algoritmo para extrair o ícone correto e que possua a melhor qualidade. Este algoritmo é realizado por meio dos resultados do uso do programa *wrestool*, o qual acompanha o *Porting Center*.

Primeiramente, o *wrestool* lista todos os ícones do formato GROUP_ICON que estão dentro do aplicativo EXE do qual se deseja obter o ícone (com o comando "`wrestool --raw -t14 <ARQUIVO> --output=.`"). Os artefatos GROUP_ICON são arquivos ICO inseridos dentro do EXE, mas em alguns casos o EXE pode possuir mais de um GROUP_ICON, sendo os outros designados para outros fins, como o ícone que será atribuído a um arquivo que é aberto pelo dito EXE. A figura 29 mostra o ícone de um documento Word, o qual está contido dentro do executável do Word.

Figura 29: Ícone de um documento do Word.

Quando existe mais de um GROUP_ICON, é preciso tomar uma decisão. Baseado em diversos casos, o seguinte algoritmo é adotado: Há uma variável chamada *name* para todos os GROUP_ICON, a qual pode ser numérica ou *string* (texto), e há uma chamada *size*, a qual contém o peso (tamanho) do dito GROUP_ICON. Se *name* for numérica, é obtido o GROUP_ICON cujo *name* é igual a 0 (zero), caso contrário é obtido o GROUP_ICON de maior valor em *size*, pois será o arquivo de maior qualidade.

Depois que o GROUP_ICON é escolhido, ele é salvo em formato ICO para que possa passar pelo tratamento de extração, que será explicado em 6.1.2.2 (a extração é feita pelo comando "`wrestool -x -o<DESTINO> -t14 -n<NAME> <ARQUIVO>`"). Apesar disso, há um outro caso que pode ocorrer: o EXE pode não conter GROUP_ICON algum. Neste caso, é preciso buscar por outro tipo de artefato que deve vir a se tornar o arquivo ICO: o ICON (a busca é feita via "`wrestool --raw -t3 <ARQUIVO> -o/tmp/GSexeI.ico`"). O uso do ICON é muito incomum, mas foi visto por exemplo no jogo *Stubbs The Zombie*, e por isso o Porting Center está pronto para lidar com isso.

Uma diferença presente nesse tipo de extração é que o ICO extraído de um ICON possui apenas um ícone, enquanto os GROUP_ICON normalmente possuem diversas cópias do mesmo ícone com qualidades e tamanhos diferentes. Algo que tem funcionado sempre é a escolha do ICON de maior peso, mas apesar disso há uma última adversidade:

Artefatos do tipo ICON não podem ser salvos no formato ICO diretamente e nem em outro formato, coisa que foi observada ao tentar extraí-lo de *Stubbs The Zombie* (a extração é feita pelo comando "`wrestool -x -R -o<DESTINO> -t3 -n<NAME> --language=<LANGUAGE DO ICON NA LISTA> <ARQUIVO>`"). Apesar disso ter sido um enigma no começo, foi observado que esse mesmo ícone poderia ser extraído por uma aplicação em *Java*, o *AnywherePEViewer*, para o formato ICO.

A diferença foi constatada ao abrir, em um leitor hexadecimal, o ICON extraído pelo *wrestool* lado a lado do ICO extraído pelo *AnywherePEViewer*: Havia um trecho faltante no arquivo ICON (o qual foi constatado depois que era capaz de tornar qualquer artefato ICON em um ICO). Assim, o *Porting Center* passou a inserir o trecho da figura 30 no arquivo ICO gerado antes de salvá-lo.

```
00 00 01 00 01 00 80 80 00 00 01 00 18 00 28 C8 00 00 16 00 00 00
```

Figura 30: Trecho faltante para transformar um ICON em um arquivo ICO

E isso resolveu a extração do ícone de um arquivo EXE para o formato ICO em todos os casos observados. Agora vem a segunda parte deste desafio: extrair o melhor ícone contido dentro de um arquivo ICO.

6.1.2.2 - Extração de .ICO

O formato ICO é suportado pelo programa *Pré-Visualização*, que faz parte do ambiente macOS, porém nem sempre ele consegue mostrar a imagem corretamente, pois às vezes essa imagem encontra-se criptografada, como é o caso do ícone do jogo *Mass Effect*. Sendo assim, o programa *icotool* é usado para retirar o melhor ícone de um ICO, de maneira que ele se torne um PNG que pode ser facilmente lido pelo *Porting Center*.

A escolha do ICO advém de todas as suas variáveis: height (altura), width (largura) e depth (intensidade de cor); o ícone que possuir todas as variáveis maiores ou iguais aos outros será escolhido (listagem é feita pelo comando "`icotool -l <ARQUIVO>`", e a extração do ícone escolhido via "`icotool -x -i<ÍNDICE> -o<DESTINO> <ARQUIVO>`"). Considerando que todos as imagens contidas no ICO são sempre o mesmo ícone em qualidade e tamanhos diferentes, isso sempre garante o melhor resultado. A figura 31 apresenta a listagem dos ícones presentes no maior ICO do jogo Mass Effect, é o possível observar que o ícone de maior qualidade é o de índice 10.

```
--icon --index=1 --width=16 --height=16 --bit-depth=4 --palette-size=16
--icon --index=2 --width=16 --height=16 --bit-depth=8 --palette-size=256
--icon --index=3 --width=16 --height=16 --bit-depth=24 --palette-size=0
--icon --index=4 --width=24 --height=24 --bit-depth=8 --palette-size=256
--icon --index=5 --width=24 --height=24 --bit-depth=24 --palette-size=0
--icon --index=6 --width=32 --height=32 --bit-depth=8 --palette-size=256
--icon --index=7 --width=32 --height=32 --bit-depth=24 --palette-size=0
--icon --index=8 --width=48 --height=48 --bit-depth=8 --palette-size=256
--icon --index=9 --width=48 --height=48 --bit-depth=24 --palette-size=0
--icon --index=10 --width=256 --height=256 --bit-depth=32 --palette-size=0
--icon --index=11 --width=16 --height=16 --bit-depth=32 --palette-size=0
--icon --index=12 --width=24 --height=24 --bit-depth=32 --palette-size=0
--icon --index=13 --width=32 --height=32 --bit-depth=32 --palette-size=0
--icon --index=14 --width=48 --height=48 --bit-depth=32 --palette-size=0
```

Figura 31: Listagem dos ícones presentes no maior ICO do jogo "Mass Effect". Vários ícones, um ICO

Mesmo assim, há um caso documentado que gera uma exceção: Às vezes, o *icotool* pode não ser capaz de identificar corretamente a transparência, e então toda transparência superior ou igual a 50 se torna 100, e toda inferior a 50 se torna 0. Curiosamente, quando isso ocorre, o número de cores da imagem também parece ser afetado. O resultado, como se pode observar na figura 32, é desanimador.

Figura 32: Ícone extraído pelo icotool e ícone visualizado pelo sistema de "Robust MotionDeblur"

Porém há uma saída: logo depois que a extração do *icotool* é feita, o ícone visualizado pelo sistema também é extraído, e então o peso de ambos é comparado. Quando há perda de pixels com o *icotool*, o ícone do sistema se torna mais pesado; quando o ícone do sistema é invisível, o ícone do *icotool* se torna mais pesado. Por fim, o ícone mais pesado é o escolhido.

Com isso, temos o ícone no formato PNG, o qual pode ser facilmente lido pelo *Porting Center* e transformado em um arquivo ICNS. Porém, antes desta conversão, há um problema bem comum que pode ocorrer: os ícones de aplicativos de Windows costumam ter dimensões de 32x32 a 256x256, em vez de 512x512 ou 1024x1024, que são os máximos permitidos para arquivos ICNS, dependendo da versão do Mac OS X, o que faz com que o potencial do sistema não seja aproveitado.

Para superar esse problema, foi necessário aprimorar o ícone em PNG extraído do EXE ou ICO (ou mesmo qualquer imagem que possuísse dimensões inferiores a 512x512), como explicado em 6.1.2.3.

6.1.2.3 - Aprimoramento de Imagem

Em um primeiro momento, imagens extraídas de um EXE ou ICO de uma aplicação antiga de Windows podem parecer ruins, para não dizer inutilizáveis. Mesmo assim, às vezes o usuário pode querer manter o estilo original, ou mesmo não ter uma melhor opção. Para superar isso, e para melhorar ainda mais a qualidade dos ícones que estavam próximos da qualidade máxima, foi decidido que eles deveriam ser aprimorados.

A primeira dificuldade foi saber exatamente o que procurar. Redimensionar puramente a imagem foi algo imediatamente descartado, pois todos os programas capazes de fazê-lo traziam resultados pouco satisfatórios ao ampliar imagens pequenas demais, deixando "pixealizadas" demais ou muito borradas. A figura 33 apresenta o ícone de *Commandos: Behind Enemy Lines* aprimorado com Pré-Visualização e Gimp.

Figura 33: O ícone de "Commandos: Behind Enemy Lines" original, aprimorado com Pré-Visualização e com Gimp, respectivamente

Inicialmente, foi testada a vetorização dos ícones, o que os tornaria imagens SVG de qualidade virtualmente infinita (o formato SVG será melhor explicado em 6.1.2.4). Foi utilizado o programa *VectorMagic*, porém o mesmo não produziu bons

resultados em imagens pequenas, que eram o foco principal desta funcionalidade, além de ser um software pago.

A segunda alternativa foi tentar desenvolver essa funcionalidade no Porting Center, mas isso mostrou-se rapidamente inviável, pois exigiria muitos conhecimentos sobre como a imagem deveria se formar depois que fosse estendida, os quais envolviam algoritmos complexos que poderiam inclusive render um projeto a parte, o que não era o objetivo.

Após mudar os termos da busca simplesmente para *"aprimoramento de imagem"*, foi descoberta a aplicação *SmillaEnlarger*. Com sua versão pré-compilada para Mac, o *SmillaEnlarger* provou ser sem dúvida a escolha certa, devido aos impressionantes resultados e à fácil integração com o Porting Center.

As imagens resultantes não foram impressionantes apenas nas imagens pequenas, mas também transformaram imagens grandes em ícones perfeitos para uso em aplicações. Na figura 34 é possível ver uma imagem aprimorada pelo *SmillaEnlarger*, onde a imagem resultante tem contornos bem mais definidos.

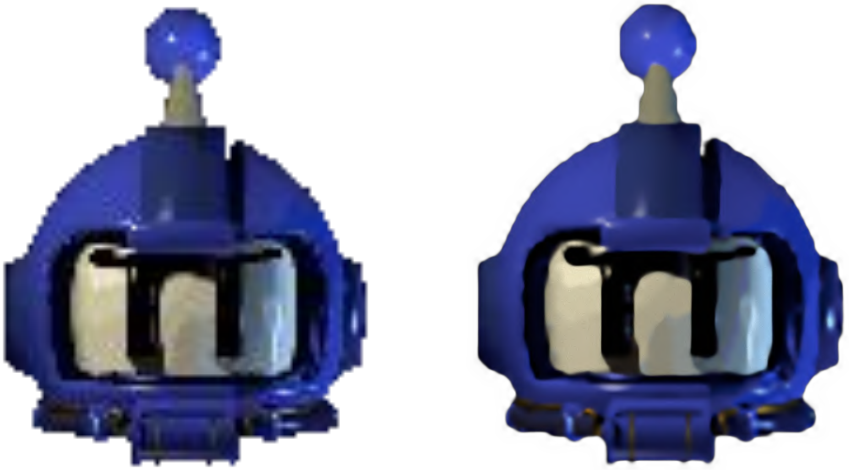

Figura 34: Ícone original baseado em Bomberman e ícone aprimorado pelo SmillaEnlarger, respectivamente

No entanto, ainda havia mais um obstáculo a ser superado: a ausência de contorno vazio. Quando o *SmillaEnlarger* aprimora uma imagem em que há pixels não-transparentes nas linhas e colunas que contornam a imagem (o que é muito comum nos ícones pequenos devido ao pouco espaço disponível para a imagem),

ele deduz sempre que a imagem deveria continuar para além do espaço do ícone, o que estraga o resultado. Isso pode ser observado na figura 35.

Figura 35: Ícone de "Commandos: Behind Enemy Lines" aprimorado sem contorno vazio

Devido a esses casos, o Porting Center verifica se existe um contorno transparente ao redor da imagem antes de aprimorá-la, e caso não haja ele o insere. Isso diz ao *SmillaEnlarger* que imagem está completa, e que é um corpo independente, como mostrado na figura 36.

Apesar disto ser considerado como regra geral, pode ocorrer um caso em que uma imagem realmente seja contínua para fora da área do ícone em uma ou mais direções, mas como nenhum caso assim foi visto ainda nos testes, esse método se mantém como o mais eficaz e de melhor resultado.

Figura 36: Ícone de "Commandos: Behind Enemy Lines" aprimorado com contorno vazio

Porém, existe uma exceção que ainda precisa ser atendida: quando o contorno da imagem é completamente visível (não-transparente) pode significar que

a intenção da imagem é realmente cobrir toda a extensão do ícone, e neste caso o Porting Center deve permitir que o *SmillaEnlarger* aprimore sem o contorno transparente. No exemplo da figura 37, envolver toda a conjectura do ícone pode ter sido a intenção do autor da imagem.

Figura 37: Ícone da empresa Lionhead Studios extraído do jogo "Black & White"

Como este é um exemplo que varia com os casos, talvez o Porting Center conte futuramente com a possibilidade do usuário escolher se deve ou não ter a borda inserida, assim como uma escolha automatizada. Com tudo isso, as imagens pequenas podem ser muito bem aproveitadas.

Muito tempo depois, descobri que o *SmillaEnlarger* possuía dependências que deveria estar instaladas no Mac ou virem acompanhadas do programa, as quais consumiam cerca de 33.8Mb (que é muito mais pesado que o próprio programa). Assim sendo, a função de aprimoramento do *SmillaEnlarger* passou a ser opcional, funcionando apenas se o mesmo já estiver instalado no computador.

6.1.2.4 - Rasterização de .SVG

O formato SVG está associado a imagens vetorizadas, as quais são imagens que são *desenhadas* no momento da visualização a partir de pontos definidos em um arquivo derivado do XML. Isso faz com que a qualidade das imagens SVG seja denominada como *virtualmente infinita*.

A rasterização é o processo pelo qual uma imagem vetorizada se torna uma imagem comum, podendo assim ser usada e visualizada por aplicações tradicionais. Um recurso muito útil, sem sombra de dúvidas.

Figura 38: Imagem SVG de logotipo da companhia Apple após rasterização

Obviamente, imagens com esse grau de qualidade não poderiam ficar de fora dos formatos suportados para ícone. A qualidade dos programas no ramo era sempre equivalente, por isso o foco ao procurar um conversor deveria ser velocidade e leveza; ou seja, o que realizasse a conversão mais rapidamente e sem consumir muita memória.

O primeiro teste veio com um programa chamado *batik-rasterizer*. Sua conversão levava aproximadamente 3 segundos, mas consumia 3.2 Mb, o que era mais do que o consumo do próprio Porting Center (na época, ainda Wigidrasil). Além disso, o fato de ele ser feito em Java era problemático, pois o Mac OS X removeu o suporte nativo ao Java na versão 10.7, o que exigiria que o Java fosse baixado manualmente para usar um único recurso. Além disso, nem todas as imagens eram suportadas.

Logo em seguida, em um teste feito puramente para avaliar a velocidade de processamento, foi constatado que o programa *Inkscape* poderia realizar a conversão via linha de comando, mas apesar de levar menos de 1 segundo para fazê-lo, ele consumia 299.7 Mb em memória, o que o dispensava automaticamente. Além disso, a questão do tamanho da imagem também era problemática nele.

O *qlmanage* veio depois. Por fazer parte do sistema do Mac, tinha a vantagem de não acarretar nenhum consumo de memória extra, porém suas conversões sempre incluíam um fundo branco, o que tira todo o valor das imagens que possuem um fundo transparente, que são a grande maioria. Além disso, o *qlmanage* não era capaz reposicionar as coordenadas dos pontos de acordo com um tamanho especificado, o que tornava o resultado muito pequeno e de pouca qualidade. Apesar disso, levava menos de 1 segundo para realizar a conversão.

Depois de muita busca, foi utilizado o *rsvg-converter*, o qual se provou perfeito para o caso. Não apenas pelo fato de consumir apenas 1.4 Mb de memória, como também foi capaz de realizar uma conversão de SVG para PNG em aproximadamente 2 segundos nos testes realizados. A comparação das ferramentas usadas na imagem da figura 26 em um mesmo computador podem ser vistas na tabela 3.

	Tempo	**Consumo**	**Qualidade**	**Sucesso**
batik-rasterizer	1,123 s	3.2 Mb + Java	Ótima	Às vezes
Inkscape	0,581 s	299.7 Mb	Média	Sempre
qlmanage	0,169 s	0 Mb	Baixa	Sempre
rsvg-converter	~ 2 segundos	1.4 Mb	Ótima	Sempre

Tabela 3: Comparação das ferramentas usadas para rasterizar arquivos SVG

Conclusão: sustentando-se principalmente no consumo de memória e na qualidade da imagem, a escolha foi o *rsvg-converter*, pois valia a pena levar um pouco mais de tempo na conversão do que ter um grande consumo adicional de memória ou uma qualidade pior da imagem.

No entanto, depois de um certo tempo, descobriu-se que o rsvg-converter necessitava de diversas dependências que precisavam estar instaladas no computador. Para não remover esse recurso, ele passou a estar disponível apenas para àqueles que possuem o *rsvg-converter* já instalado. Se ele não estiver presente o *Inkscape* pode ser usado no lugar.

6.1.3 - Obtenção de Dados de .EXE

Muitas vezes pensamos nos arquivos EXE apenas como arquivos executáveis de Windows e nada mais, porém eles são muito mais do que isso. Não me refiro apenas ao ícone (o qual foi mostrado na seção 6.1.2), mas a informações textuais.

Nos itens a seguir, veremos algumas das informações que podem ser obtidas a partir de um arquivo EXE, as quais podem ajudar a tornar um port ainda mais autêntico, ou mesmo auxiliar em sua criação. As informações que podem ser obtidas são divididas em dois grupos: as visíveis e as invisíveis.

As visíveis são aquelas que, estando no sistema operacional Windows, podem ser visualizadas simplesmente abrindo as propriedades do EXE (Nome/Versão/Copyright). As invisíveis são aqueles que, independente do sistema, não deveriam ser passíveis de visualização, mas não quer dizer que não possam ser achadas (como por exemplo, as bibliotecas necessárias).

6.1.3.1 - Obtenção de Nome/Versão/Copyright

Além do ícone, 3 variáveis de um aplicativo são mostradas pelo macOS durante o *QuickLook* (recurso do Mac que revela dados básicos de um arquivo ou seu conteúdo na íntegra): Nome, Versão e Copyright. O nome é algo simples, mas a versão e o copyright nem sempre são algo trivial para o usuário saber. Na figura 39 temos uma janela do QuickLook de port para o jogo *Phantasmat* com dados obtidos pelo Porting Center em uma versão mais antiga.

Figura 39: QuickLook de port do jogo Phantasmat

O Nome costuma ser somente o nome pelo qual o aplicativo é conhecido (Ex.: Phantasmat). A versão é normalmente um número em ponto flutuante que corresponde à versão do aplicativo (Exemplo: 1.0). E Copyright é a junção do ano de criação do software com a empresa que desenvolveu o programa, e a frase *"All rights reserved."* no final (Exemplo: Copyright © 2011 Codeminion. All rights reserved.).

Normalmente esses dados deveriam ser preenchidos manualmente pelo usuário, mas o fato é que eles estão contidos dentro da grande maioria dos arquivos EXEs para *Microsoft Windows*. A questão é apenas: como extrair essas informações?

Durante os testes com o *wrestool* para extração de ícones de EXE (mais detalhes em 6.1.2.1) notei que havia quase sempre a presença de uma variável chamada DETAILS. Extrai essa variável para um arquivo de texto (pelo comando "`wrestool -x -t16 -n1 -R <ARQUIVO> -o<DESTINO>`", ou "`wrestool -x -t16 -R <ARQUIVO> -o<DESTINO>`" se o anterior falhar), a abri com um editor de texto convencional (*TextEdit*), e então me deparei com estas informações. A partir daí, pensei em extraí-las e usá-las sempre que possível.

Assim que isso foi cogitado, abri o arquivo no editor hexadecimal *Hex Fiend* para procurar a presença de padrões que pudessem levar aos textos em suas devidas partes do arquivo, como foi feito para descobrir as bibliotecas necessárias, que será explicado no item 6.1.3.2. Logo percebi que haviam caracteres nulos

(Valor hexadecimal: 00) entre cada dígito do arquivo (normal, nulo, normal, nulo, e assim sucessivamente).

Além disso, antes de cada informação provida pelo arquivo havia um nome identificador, no entanto mesmo esse nome possuía os ditos caracteres nulos, o que dificultava a criação de uma lógica capaz de encontrar esses nomes e obter os valores após eles. A razão pela qual era possível ler com o arquivo com o *TextEdit* é por que ele mostrava o caracter nulo como um caractere vazio e de largura zero, ou seja, “invisível".

Para superar isso, o programa passou a ler o arquivo em hexadecimal e a remover todos os caracteres em índices pares antes de começar a trabalhar, removendo assim os caracteres nulos que impossibilitavam a busca pelos nomes identificadores. Logo depois disso, ele passa a buscar pelos nomes identificadores no arquivo para encontrar as informações.

Tanto 'Nome' quando ‘Copyright’ possuem mais de um identificador que pode ter a informação desejada, e ‘Versão' só possui um identificador que pode ter sua informação. Isso foi observando extraindo o DETAILS de vários executáveis diferentes. Caso alguma dessas três informações não seja encontrada, existe um substituto (ou *placeholder*) que pode ser usado. Os identificadores e *placeholders* usados podem ser observados na tabela 4.

	Identificador 1	**Identificador 2**	**Placeholder**
Nome	*ProductName*	*FileDescription*	Nome do EXE
Versão	*FileVersion*	—	1. 0
Copyright	*LegalCopyright*	*CompanyName*	“*Copyright ©*” + Ano de criação + 2ª pasta do caminho do arquivo + “All rights reserved.”

Tabela 4: Lista dos identificadores e placeholders por informação

Se em nenhuma das tentativas for encontrado o dado, o programa *desistia* e entregava um dado que será no mínimo útil ou aceitável por falta de opções. Por

exemplo: se um dito arquivo *“C:/Arquivos de programas/Eidos Interactive/Pyro/Commandos/COMANDOS.exe”*, não possui DETAILS e foi criado em 1998, seu dados eram respectivamente: ‘*COMANDOS’*, ‘*1.0’*, e ‘*Copyright © 1998 Eidos Interactive. All rights Reserved.’*.

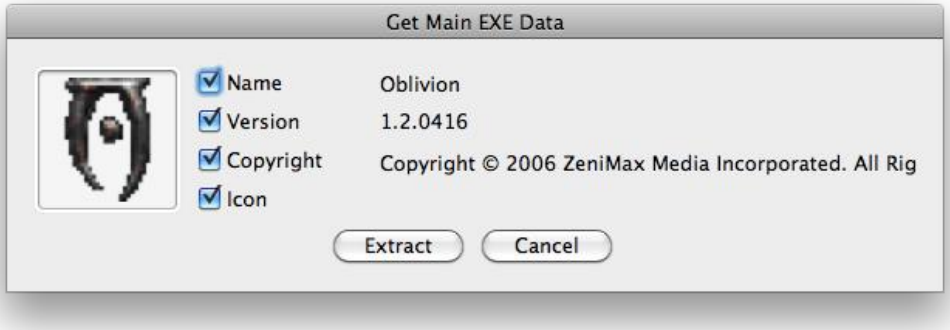

Figura 40: Tela de extração de dados de EXE do jogo “The Elder Scrolls IV: Oblivion”

Como esses três dados podem ser extraídos do arquivo EXE pelo Porting Center, o usuário não precisa mais obter esses campos, e assim economizar tempo escrevendo (e às vezes até pesquisando) o que eles deveriam conter.

Infelizmente, conforme esse recurso foi usado, percebi que a maioria das empresas não se preocupava com o versionamento presente no executável, e às vezes até o nome do programa era apresentado de forma incorreta ou incompleta, como *The Elder Scrolls V: Skyrim*, em que o nome do executável principal era *Skyrim Launcher*. Devido a isso, esse recurso não foi movido para o Porting Kit durante a transição de maio de 2017.

6.1.3.2 - Descobrir bibliotecas necessárias

Quando se está tentando instalar certos programas, algumas bibliotecas podem ser necessárias para que ele seja executado (principalmente quando se trata de jogos, com bibliotecas como *DirectX* e *PhysX*), as quais devem ser instaladas por meio de winetricks, métodos que redirecionam as chamadas que vão para as bibliotecas para que se encaminhem a funções equivalentes do sistema nativo.

O grande problema é que o usuário comum não sabe quais são as bibliotecas que um programa específico vai precisar (e até mesmo um usuário mais experiente costuma ter sérios problemas com isso) e muito menos quais os winetricks que ele vai precisar para resolver isso.

Quando o *Porting Center* ainda era *Appleverse Wizard*, abri acidentalmente um arquivo EXE com um editor hexadecimal, enquanto fazia testes para extração de imagens de arquivos ICO (mencionadas em 6.1.2.2), e então notei que os nomes das DLLs que eram chamadas estavam escritas em texto puro.

Notando isso, criei um algoritmo em Java que era capaz de detectar a presença das strings *.dll*, *.ocx* e *.ax*, que são os três tipos de extensão comuns para bibliotecas no Windows. Ao detectar as strings, o programa copiava os 25 dígitos que viessem antes delas, e a partir de uma análise destas strings chegava ao nome da DLL.

Apesar de ser um recurso muito útil o algoritmo era muito lento, e se o EXE fosse grande demais o programa praticamente travava, e então não havia como continuar. Quando o programa se tornou *Wigidrasil* e foi passado para *Objective-C*, essa funcionalidade ficou um certo tempo fora do programa.

Quando chegou o momento de reimplementá-la, o código foi refeito completamente, porém desta vez na linguagem *C*. O resultado foi surpreendentemente melhor, levando menos de 1 segundo para extrair os nomes de DLLs de executáveis, antes considerados grandes demais para abrir.

Assim que extraía o nome das bibliotecas, o Porting Center verificava uma lista que possuía, e então mostrava quais eram os winetricks necessários para a instalação das bibliotecas encontradas. A figura 41 mostra a tela do Porting Center apresentando essa lista para o executável de *The Elder Scrolls IV: Oblivion*.

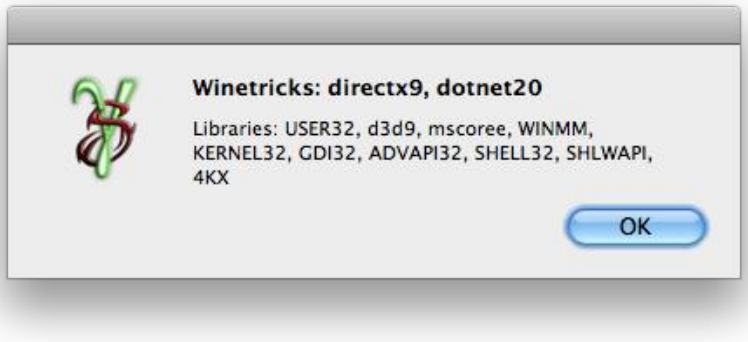

Figura 41: Janela original de descoberta de bibliotecas/winetricks necessários

Lembrando que apesar das bibliotecas estarem corretas, os Winetricks mencionados são apenas sugestões baseadas nos resultados, mas nem sempre eles são realmente necessários, e em alguns casos algo mais simples pode ser suficiente. No caso apresentado na figura 41, o mencionado executável só precisou de um winetrick chamado *d3dx9_36* (derivado do *directx9*) para que pudesse funcionar no motor *WS9Wine1.3.18*.

Apesar de ser extremamente mais rápido que o Appleverse Wizard, o Porting Center ainda não possuía o mesmo design que ele para esta janela, e isso levou muito tempo para ser implementado.

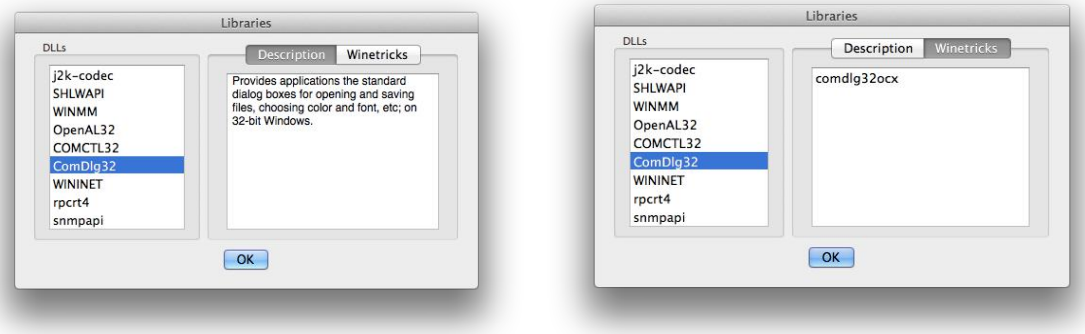

Figura 42: Nova janela com Descrição e Winetricks por Biblioteca

Esta parte do programa atualmente é bem similar à sua versão original no Appleverse Wizard em matéria de design, tendo sido aprimorada inclusive, sendo capaz de analisar a lista de winetricks disponível no wrapper e descobrir quais as bibliotecas em que ele realiza sua instalação, como mostrado na figura 42.

A próxima e última etapa desse sistema seria permitir que os Winetricks fossem instalados diretamente desta janela, no entanto, como mencionado anteriormente, isso acabaria instalado coisas desnecessárias.

Quando o Porting Center e o Porting Kit se combinaram esse recurso ficou de fora, com planos para que possa retornar no futuro, mas desta vez apenas dizendo quais as bibliotecas necessárias para reproduzir um certo executável.

6.1.4 - Obtenção de Caminho

Os wrappers sempre possuem um campo que define qual arquivo será executado por ele. Por exemplo, um programa de Windows, após sua instalação, resulta em diversos novos arquivos, normalmente localizados na pasta de Arquivos de Programas.

No entanto, apesar de haver dezenas, ou até centenas de arquivos gerados, apenas um é utilizado pelo usuário: o executável. Ele é quem chama os outros arquivos, os executa, os lê, ou o que quer que tenha que fazer com eles. Justamente por isso ele é o único arquivo que possui um atalho na área de trabalho.

Já no Mac OS X, o normal é que um programa instalado resulte em um único arquivo, que na realidade não é bem um arquivo. Os .app na realidade são pacotes, o que nada mais são do que pacotes (diretórios) capazes de serem executadas por possuírem um caminho de execução pré-escrito em um arquivo PLIST (variação do XML) que se encontra dentro dos .app sempre no mesmo local e com o mesmo nome.

Nos ports dos mais variados tipos, o caminho definido no .app é para um binário capaz de portar programas de Windows, no entanto existe um segundo campo onde se armazena qual executável do Windows esse binário deverá executar. Isso converte o conceito de atalho em um conceito de execução direta, o que trás ao usuário de Mac a mesma comodidade à qual já está acostumado.

6.1.4.1 - Extração de .LNK

Os arquivos de extensão .lnk são usados no sistema operacional Windows como caminhos mais ágeis para alcançar outros arquivos, conhecidos normalmente como *atalhos*. Depois que um software de Windows é instalado, é gerado um atalho na Área de Trabalho, e nesse caso o usuário não precisa se preocupar em saber onde o executável está, e nem quais flags ele precisa.

Para dar essa mesma comodidade a quem usa wrappers, a única solução seria extrair o caminho desses arquivos LNK, os quais são gerados durante a instalação de um software em um wrapper. A questão é apenas como fazer isso.

No decorrer do tempo, a estrutura interna dos arquivos LNK foi sendo modificada, e isso pode ser observado abrindo atalhos em um editor hexadecimal, dentre eles alguns vindos de computadores com Windows XP até o Windows 8. Observando isso, foi preciso procurar o que havia de comum nos dois.

Antes de começar, é preciso lembrar que às vezes existem espaços com hexadecimal nulo (valor 00) entre cada dígito de um caminho no arquivo LNK, por isso ele é analisado três vezes: uma inteira, outra apenas com os dígitos pares, e a última apenas com os dígitos ímpares.

O nome do executável sempre acompanha o resto do caminho (*path*) ao menos uma vez, então é este caso que o algoritmo procura. Apesar disso, de vez em quando, o resto do caminho está incompleto, faltando a letra do drive do caminho. Para esses casos, existe um algoritmo voltado para descobrir qual o drive do arquivo citado no atalho. Por fim, um outro algoritmo é capaz de obter as flags, as quais certas vezes são necessárias para a execução do aplicativo.

6.1.4.2 - Extração de .DESKTOP

Algo muito comum durante a instalação de aplicações em wrappers Wineskin é o surgimento de arquivos da extensão DESKTOP na Mesa (nome da *Área de Trabalho* do ambiente macOS). Como um usuário de Mac, eu não conseguia compreender por que eles surgiam, mas com o uso do sistema operacional Linux Mint na universidade percebi a sua finalidade.

O código fonte do Wine é originalmente feito para o Linux, o que nos leva ao fato de que certos confortos foram criados para facilitar a vida dos usuários do mesmo. Os arquivos DESKTOP geram atalhos nos menus das distribuições Linux, fazendo com que executem os arquivos LNK que foram gerados por um instalador no Wine. Em outras palavras: são atalhos para atalhos.

No entanto, esses arquivos não possuem finalidade no Mac OS X, tornando-os *deletáveis*. Sabendo do que se tratavam, percebi que poderiam ser aproveitados: extraindo de um arquivo DESKTOP o caminho que leva até seu respectivo arquivo LNK, é possível realizar a extração de caminho do arquivo LNK que o usuário precise encontrá-lo.

Para isso, basta obter o caminho especificado no campo exec e extrair o trecho necessário. Isso se deve ao fato de que o campo costuma possuir muitas informações não úteis para esse caso, sendo ele usualmente:

"env WINEPREFIX="<Caminho do Wrapper a partir da raiz>/Contents/Resources" wine C:\\\\windows\\\\command\\\\start.exe /Unix <Caminho do arquivo LNK a partir da raiz>"

A única exceção encontrada foi para os arquivos DESKTOP gerados por meio da aplicação *Steam*. Ao requisitar a aplicação que criasse um atalho, ele gerava um arquivo que possui no lugar do caminho do LNK um *URI Scheme*, no caso:

steam://rungameid/<Número do App na Steam>

Neste caso, foi obtido o número do app, e definido o seguinte como o caminho de execução:

"C:/Program Files/Steam/steam.exe -applaunch <Número do App na Steam>"

O que é o equivalente ao *URI Scheme* na forma de um caminho de arquivo com flags. Tudo isso facilita muito o trabalho do usuário para descobrir o caminho do app desejado.

6.1.4.3 - Extração de .INF

Algumas aplicações necessitam que uma outra mídia, normalmente disco, seja totalmente copiada para dentro do wrapper para garantir o funcionamento da aplicação. Como pude observar algumas vezes, certas aplicações são executadas diretamente dessas mídias, o que no Windows seria chamado pela execução do arquivo autorun.inf.

Arquivos com essa extensão são utilizados no Windows para realizar a execução automática de um programa quando um dispositivo externo ou mídia se conecta, com o arquivo incluído dentro do próprio. A execução de apps diretamente desse tipo de dispositivo se tornou algo pouco comum, no entanto é necessário devido às aplicações mais antigas.

O arquivo autorun.inf possui uma variável OPEN, a qual possui o caminho para o arquivo .EXE, a ser executado a partir do diretório do próprio arquivo *autorun*, que normalmente estaria na raiz de um disco por exemplo.

6.1.5 - Controle de Drives

Durante a criação de ports, me deparei mais de uma vez com jogos antigos que precisavam que seus discos estivessem inseridos no computador para serem executados, ou ao menos que houvesse um drive no port igual ao seu disco.

Como foi mencionado antes, cada wrapper possui seus próprios drives, o que é algo favorável por diversos fatores, por exemplo caso o usuário não tenha certeza se o arquivo que quer executar está livre de vírus, ou se quiser instalar uma aplicação que precise de seu CD/DVD inserido.

Considerando essa possibilidade (além de outras mais), o usuário pode pensar que uma área de controle de drives é algo comum entre os utilitários de

wrappers. No entanto, nenhum dos utilitários possui esse recurso de maneira explícita. Ao tentar criar um drive, é necessário executar uma aplicação interna do wrapper para então manusear os drives de uma maneira muito complexa para o usuário leigo.

Para superar isso foi criado um wrapper e foi feita uma cópia do mesmo. No original foi inserido o drive e na cópia não, e então os dois foram comparados arquivo por arquivo, procurando por diferenças entre os dois onde os drives poderiam estar configurados.

A conclusão foi que isso tudo era definido em dois locais apenas: os *alias* (atalhos do Mac) na pasta *dosservices* que definiam em que pasta estavam os drives, e algumas linhas no registro (*system.reg*) do wrapper, que diziam de que tipo era cada drive. Editar esses arquivos significava controlar todos os drives do port.

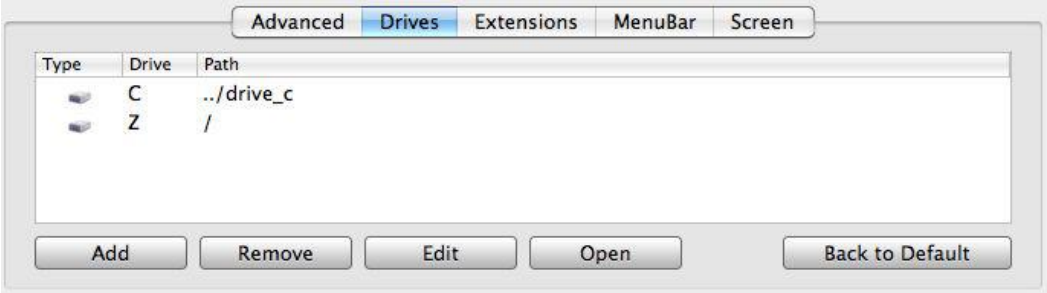

Figura 43: Aba de Drives no Porting Center

Um drive pode ser de cinco tipos diferentes: Automático, Disco Rígido, CD-ROM, Disquete e Pasta Compartilhada, sendo Automático e CD-ROM os mais utilizados. Esse tipo é armazenado no registro, que é um dos locais editados pelo programa.

Para isso foi criada uma aba no Porting Center, capaz de gerenciar os caminhos destes *alias* e modificar as informações necessárias no registro. Essa aba pode ser observada na figura 43.

Para a mudança no tipo de drive foi criada uma janela a parte, acessada ao se pressionar 'Edit' com um drive selecionado. Nela, o usuário pode modificar qual o tipo do drive selecionado. Essa janela está presente no figura 44.

Figura 44: Janela de troca de tipo de drive no Porting Center

Quando foi passado para o Porting Kit, a janela mostrada na figura 44 foi substituída por um combo box. A ideia era simplificar o uso desta janela para o usuário mais leigo, que poderia não entender o objetivo do botão *Edit*.

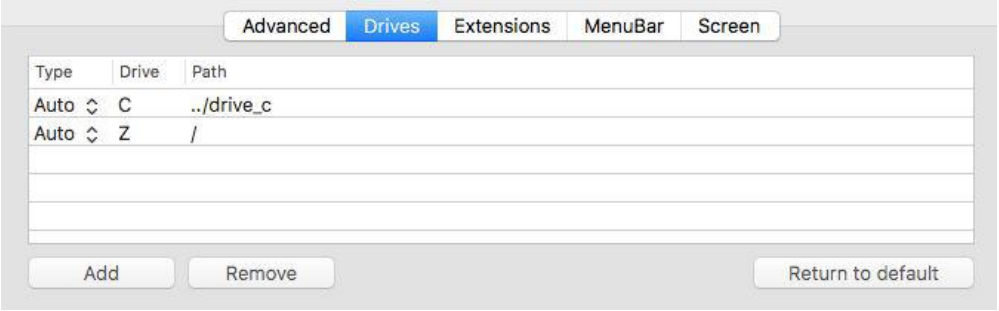

Figura 45: Aba de Drives no Porting Kit

6.1.6 - Edição de Menu

Uma coisa que nunca muda em relação a aplicativos de Mac é que por mais simples que sejam, eles possuem um menu. Isso não poderia ser diferente com os ports. Em ports Wineskin é possível inserir seus próprios itens no menu, porém de uma maneira manual e nada simples para iniciantes.

Depois de incluir as cinco linhas padrões do Wineskin que preparam o ambiente, o usuário deve inserir uma linha de comando do Wine com a sua instrução, o que não é algo trivial para um usuário inexperiente. Depois de compreender o funcionamento das abas extras de menu, fiz com que o Porting Center fosse capaz de gerar esses arquivos, como apresentado na figura 46, onde

‘Menu' define em qual aba o item deve estar presente, em ‘Item' se define seu nome e em ‘Function' a instrução do Wine que deve ser utilizada por ele.

Um “efeito colateral” presente foi que o wrapper era incapaz de reconhecer os menus gerados pelo Porting Center, mas isso logo se mostrou ser uma falha nas permissões do arquivo gerado, problema que foi resolvido com uma linha de comando enviada ao terminal pelo programa (`chmod 755 <ARQUIVO>`).

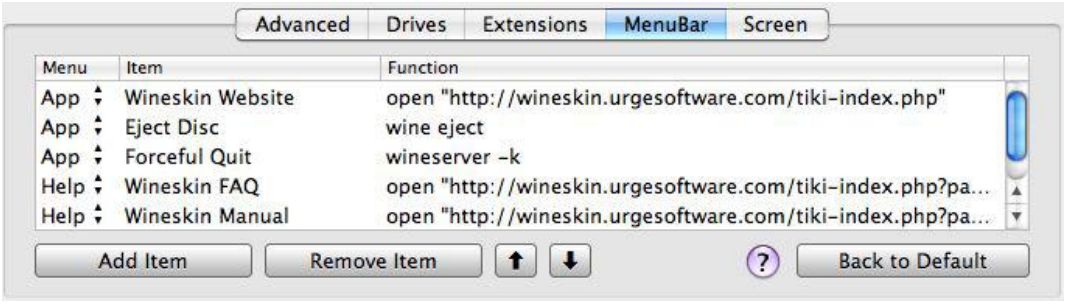

Figura 46: Aba de MenuBar no Porting Center

Para permitir que usuários mais avançados tenham máximo proveito, a função é inserida como uma linha de comando, porém ainda assim esta parte está mais fácil para os novatos, pois pressionando o botão de interrogação ("?") a esquerda do botão *Back to Default* é possível visualizar quais as funções mais comuns em uma janela, a qual pode ser vista na figura 47.

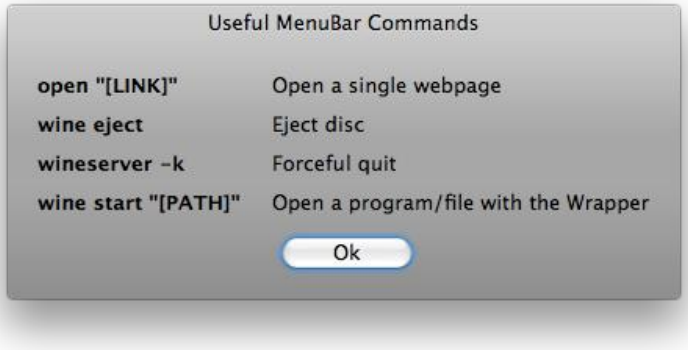

Figura 47: Janela com comandos úteis para botões do menu no Porting Center

6.1.7 - Configuração de Tela

Quando um port é executado, uma das primeiras coisas carregadas são as suas configurações de tela. Elas definem se o port terá tela cheia ou não, se terá um Desktop Virtual ou não, qual será a resolução, quantas cores terá, e se as janelas terão o design do Mac OS X ou não.

Com tantas coisas para se configurar, você deve imaginar que os outros utilitários fizeram janelas simples para esta função, certo? Errado de novo. Todas são confusas, mesmo havendo tão poucas coisas para se decidir. Considerando isso, decidi que o Porting Center deveria possuir uma janela amigável ao usuário, como visto na figura 48. A *opção Window Mode* define se serão usadas as configurações de tela passadas pelo programa de Windows, ou se o port deve ser forçado a outra coisa.

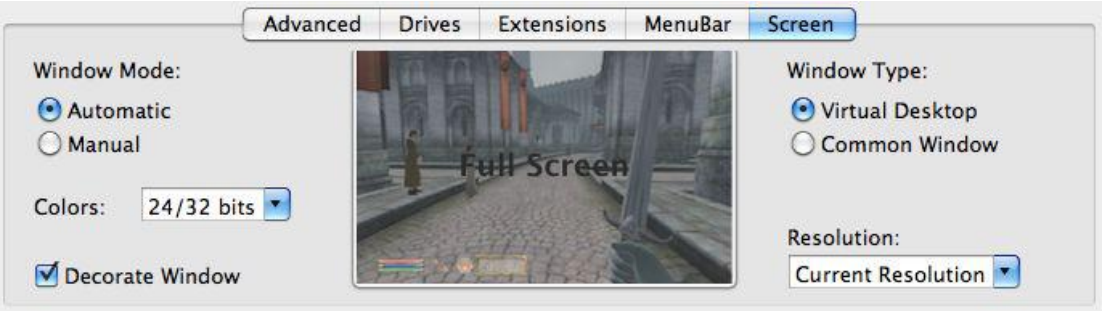

Figura 48: Aba de Tela do Porting Center

A janela teve vários problemas durante o seu processo de criação devido a todas as possibilidades de combinação de vários atributos, mas isso tudo acabou sendo solucionado. "*Resolution*", "*Window Type*" e "*Colors*" são armazenados na chave "*Resolution*" do arquivo plist do port (`<resolução ou novd>x<cores>xsleep0`), "*Window Mode*" é armazenado em "*Use RandR*" (true/false), "*Full Screen/Windowed*" em "*Fullscreen*" (true/false) e *"Decorate Window"* em "*force wrapper quartz-wm*" (true/false). O que deveria ou não ser editado foi descoberto com o mesmo método usado para descobrir o que era mudado ao inserir um drive (visto no item 6.1.5).

Algo que foi notado durante o port do jogo *Moto Racer* foi que os utilitários ofereciam apenas a coloração em 32 bits (24 bits de cores + 8 bits de alfa/vazios), enquanto o Wine, que era a raiz de todos os ports, oferecia suporte também a 8 e 16 bits.

Visando essa situação, o suporte a diferentes cores foi inserido no Porting Center, permitindo o revezamento entre essas 3 opções (8 bits/256 cores, 16 bits e 24/32 bits). Assim sendo, o usuário pode facilmente configurar o seu wrapper, o que aumenta suas chances de sucesso dependendo do aplicativo. Um exemplo é o jogo *DC Universe Online*, cujo port só funciona com Desktop Virtual e em tela cheia; outro exemplo é o jogo *Moto Racer*, que só funciona em modo de 16 bits de cores.

Com o tempo, observou-se que havia uma combinação de fatores inválida que essa janela possibilitava. A tela cheia só funciona com desktop virtual, então usando *Common Window* e *Fullscreen* o resultado seria imprevisível. Para evitar isso, durante a migração desse recurso para o Porting Kit, as opções *Full Screen/Windowed* e *Virtual Desktop*/*Common Window* foram substituídas por *Virtual Desktop* / *Common windows* / *Fullscreen*, pois o nome Virtual Desktop já está associado à sua combinação com o Windowed, em que esse nome fica visível na janela do programa.

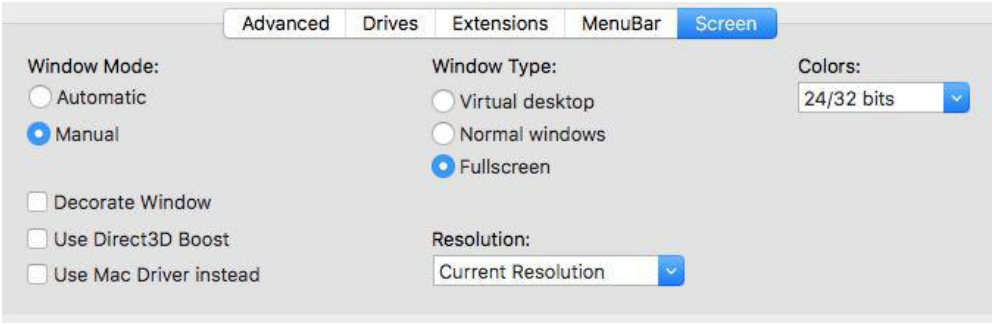

Figura 49: Aba de Tela do Porting Kit

Somando isso ao fato de que o botão *Windowed* / *Fullscreen* não era tão intuitivo como se imaginava, a aba sofreu uma boa remodelação, como pode ser vista na figura 49. Posteriormente, observou-se que as configurações de janela só funcionavam se o modo de renderização de janelas fosse o X11; usar o Mac Driver inutilizava as demais, com exceção apenas do Boost de Direct3D.

6.1.8 - Detalhes de Port

Um problema muito comum ao se baixar um port feito por outra pessoa na internet é saber se ele é do tipo que precisa de arquivos externos (como o CXZ) ou, caso o usuário seja um desenvolvedor de ports, com qual motor, sistema e winetricks aquele wrapper foi feito.

Mesmo para o usuário avançado pode não ser simples distinguir os diferentes tipos de wrapper, portanto para o usuário inexperiente isso é um tarefa quase impossível. O tipo de wrapper era definido pela presença ou ausência de certos arquivos. A versão do wrapper, assim o nome de sua engine, mudavam de local dependendo do tipo de wrapper, fazendo com que fosse necessário saber onde encontrar essas informações nos mais variados tipos de port. Por fim, o sistema operacional só poderia ser descoberto se o registro do wrapper (*system.reg*) fosse checado manualmente, ou se o *winecfg*, janela de configurações do Wine, fosse acessado.

Como saber de todas essas informações sem ter que checar diferentes locais do port e em pouco tempo? Um ser humano pode levar alguns minutos para conferir tudo, mas um programa de computador pode fazer isso em menos de um segundo. Sendo assim, tornei isso um recurso padrão do Porting Center.

Assim que o usuário abria um port no Porting Center, todos esses dados poderiam ser acessados a partir da aba *Port*, como mostrado na figura 50. Com essas informações, obtidas do port pelo cruzamento de várias linhas do plist, do Wineskin interno, do arquivo *system.reg* e de logs, o usuário iniciante poderia facilmente aprender as diferenças entre os diferentes tipos de wrappers, e aprender a criar os seus próprios baseado nos resultados de outros *portadores*.

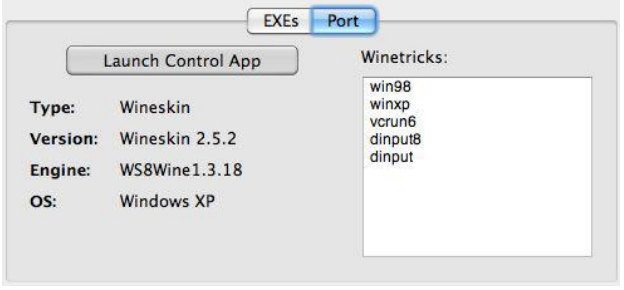

Figura 50: Aba de Port no Porting Center

E, caso o usuário precisasse abrir o utilitário original do wrapper por alguma razão em especial, o botão “*Launch Control App*” estava a disposição, permitindo que o usuário o abrisse com apenas um clique, em vez de precisar abrir o wrapper como uma pasta e então executar o programa interno do utilitário.

Apesar disso, quando o Porting Center teve seus recursos transferidos para o Porting Kit, esse dado passou a não ser mais exibido, já que os outros tipos de wrapper além do Wineskin já não eram mais compatíveis com o macOS. Mesmo assim, o Porting Kit é capaz de detectar o tipo do wrapper, e com isso pode criar um WSI deste wrapper (como será explicado mais adiante) e por meio deste criar uma versão Wineskin do antigo wrapper.

6.1.9 - Gerenciamento de EXEs

Em alguns ambientes operacionais é muito comum encontrar pacotes com vários programas, como o Office por exemplo, que contém Word, Excel e PowerPoint na sua versão mais simples. No entanto, se um port só pode ter um executável principal, como abrir um desses programa apenas?

Para ultrapassar essa limitação, os wrappers possuem um recurso conhecido na maioria deles como *Custom EXE*. Um *Custom EXE* é um arquivo executável de Mac que fica dentro do wrapper, mas que pode possuir um link externo para acessá-lo. Sua função é apenas executar o port com um executável diferente.

Com uma funcionalidade como essa, era de se esperar que os utilitários pudessem manusear facilmente esses *Custom EXEs*, alterando, criando e removendo-os quando desejado, mas infelizmente o processo concedido por todos eles é apenas a criação de um *Custom EXE*. Uma vez criado, o *Custom EXE* só pode ser editado ou apagado manualmente, sem a ajuda do utilitário.

Isso ocorreu ao tentar instalar o Microsoft Office 2003 pelo Wineskin. Era possível executar o Word, mas não o PowerPoint ou o Excel, e ao criar um Custom EXE para cada um deles se perdia o controle sobre eles. Era preciso apagá-los para alterar o caminho, o ícone ou mesmo o nome, ou então realizar a edição na mão.

Para evitar essa situação desgastante e agilizar o processo, foi implementada no Porting Center a possibilidade de criar diversos *Custom EXEs* em um curto espaço de tempo, assim como de poder removê-los e editá-los. Além disso, o Porting Center os coloca no mesmo nível do executável principal (figura 51), permitindo ao usuário decidir quem será o executável principal e quem será um *Custom EXE* com apenas um clique duplo.

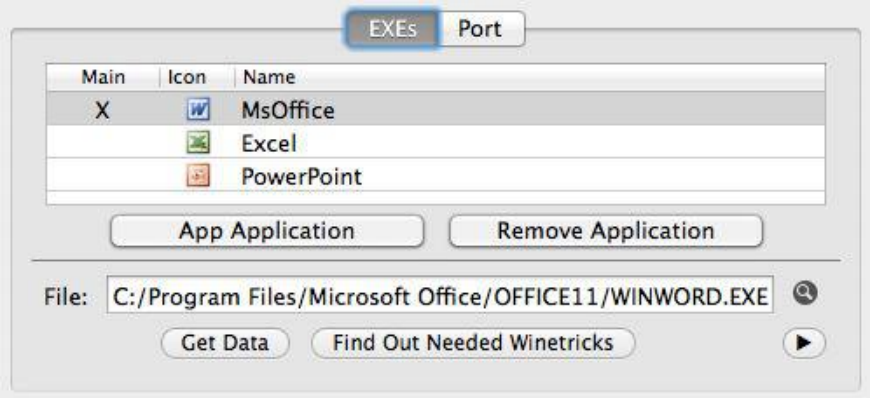

Figura 51: Aba de EXEs no Porting Center

Para fazer tudo isso, originalmente o Porting Center possuía uma cópia de *Custom EXE* de todos os tipos de wrapper, e como os mesmos não eram atualizados desde o antigo Wineskin, se mantinham os mesmos desde a criação de seus respectivos utilitários até os dias atuais. No entanto, para evitar problemas futuros e aliviar o peso do programa, ele passou a copiar o *Custom EXE* de dentro do utilitário do wrapper.

Além disso, em determinado momento, o Porting Center também passou a suportar o uso de flags nos executáveis de Windows, assim como sua extração de arquivos LNK, como já foi mencionado no item 6.1.4.

6.1.10 - Gerenciamento de Extensões

Se o usuário tem uma aplicação instalada no seu computador que é voltada para abrir arquivos de uma extensão específica (Exemplo: *WinRAR* e arquivos de extensão *.rar*), ele deseja que ao dar um duplo clique sobre um arquivo com essa

extensão, a aplicação se abra para carregar o arquivo. Com os ports, isso é uma possibilidade.

No início, não eram conhecidas as possibilidades oferecidas por esse recurso, por isso foi deixado de lado. Porém, posteriormente, ocorreram casos em que ao tentar abrir uma página de internet dentro de um port, ou um documento de texto (*.txt*) ele os abria respectivamente no *Internet Explorer* e no *Notepad* que estavam contidos no port, ao invés de no *Safari* e no *TextEdit*, navegador e editor de texto padrões do macOS.

Ao pesquisar um pouco, descobri na página oficial do Wineskin que era possível redirecionar a abertura de arquivos de certas extensões de dentro do port para o macOS, e para isso era preciso definir a função de abertura da extensão como o seguinte via Wineskin:

C:\windows\system32\winebrowser.exe "%1"

O direcionamento da abertura de extensões em um port é feito com duas variáveis: a extensão e a função, a qual define qual programa abrirá aquela extensão e de que maneira. O "*winebrowser.exe %1"* envia o caminho do arquivo que precisa ser aberto para o winebrowser.exe, o qual o encaminha para o sistema nativo, permitindo que ele seja aberto pelo programa do macOS responsável pela sua extensão.

Para permitir que esse processo seja realizado, é necessário fazer um edição no registro do port e adicionar as extensões que o port poderá abrir no arquivo Info.plist do mesmo.

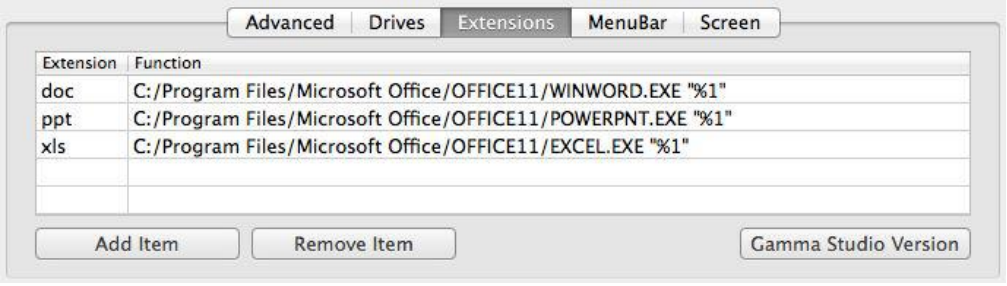

Figura 52: Aba de Extensões no Porting Center

Além disso, como pode ser visto na figura 52, também é possível configurar para que executáveis de dentro do port abram os arquivos da extensão citada, como descrito no primeiro exemplo deste sub-capítulo.

Em resumo, é possível usar esse recurso para abrir arquivos acessados pelo sistema com o port, ou abrir arquivos acessados pelo port com o sistema. É uma via de mão dupla muito útil para quem lida com vários tipos de arquivos, fazendo com que o port tenha uma “atitude nativa”.

O controle de extensões nos outros utilitários de wrappers é precário, realizado em áreas muito pequenas, necessitando normalmente de mais janelas para que se possa simplesmente trocar a função da extensão.

Esse recurso provavelmente ainda irá passar por aprimoramentos, como a possibilidade de se buscar um EXE para que se possa atribuir a função.

6.1.11 - Instalação de Winetricks

Os winetricks são essenciais para o funcionamento de muitos ports. Enquanto nem todas as DLLs do Windows foram *"recriadas"* pela comunidade Wine, é possível usá-los para instalar as bibliotecas originais do Windows dentro de um port, fazendo com que sejam interpretados e funcionem sobre funções de baixo nível já recriadas.

Apesar de não haver nenhum defeito propriamente dito na instalação de winetricks oferecida pelo Wineskin, sua UI sempre deixou a desejar por não permitir que todos os winetricks selecionados fossem visualizados simultaneamente caso estivessem distantes na lista; ou seja, se, por exemplo, os winetricks *d3dx9* e *nocrashdialog* fossem selecionados e o usuário esquecesse o que selecionou, ele teria que rever a lista inteira procurando pelas caixas que marcou.

Para evitar esse problema, o Porting Center substituiu a lista de caixas de seleção por um campo de tokens, onde cada token é um winetrick que se deseja instalar, e dar um clique duplo em um winetrick na lista o adiciona ao campo de tokens. Isso pode ser visto na figura 53.

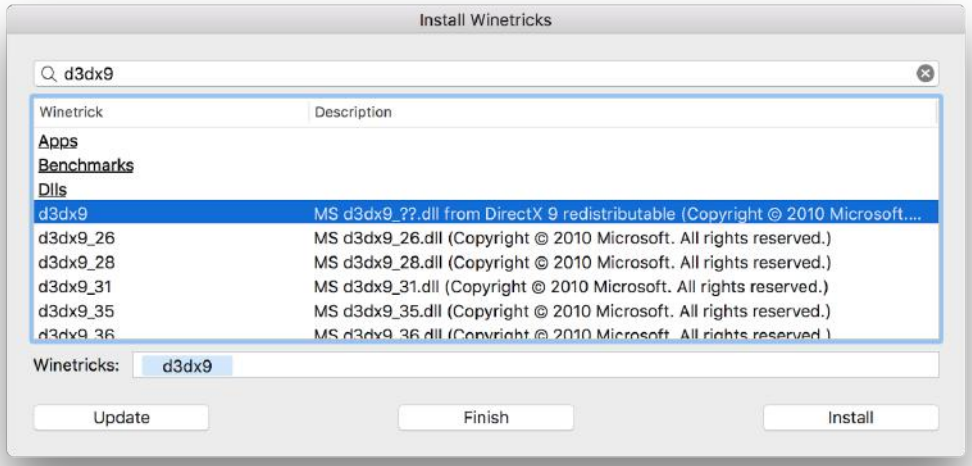

Figura 53: Janela de instalação de winetricks do Porting Center

Essa mesma janela foi passada para o Porting Kit depois de um certo tempo, com a mesma funcionalidade, com a diferença que, no Porting Kit, os logs podem ser vistos de forma colorizada, ou seja, dando ênfase a diferentes tipos de saída. Isso foi feito originalmente para facilitar o processo de *Debug* de ports.

Saídas que se iniciam com *err:* são marcadas de vermelho por indicarem erros, os quais na maioria das vezes são a raiz dos problemas de um port. Saídas que se iniciam com *fixme:* indicam “soluções” temporárias (*gambiarras*) e são marcadas de azul. Problemas também pode se originar de *fixme*’s, mas é menos comum. Saídas que não possuem nenhum desses prefixos, incluindo as que se iniciam em *wine:*, são consideradas as saídas menos relevantes, e por isso não são colorizadas. Essas outras saídas raramente são úteis. Segue um exemplo na figura 54.

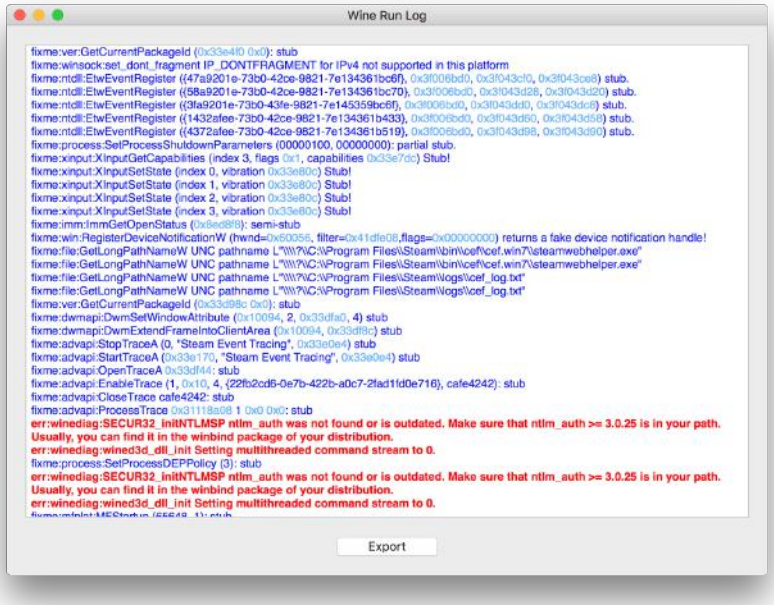

Figura 54: Visualização de log do Wine no Porting Kit

6.1.12 - Os Arquivos WSI

Este é o primeiro ponto de ligação entre o Porting Center e o Porting Kit. Durante o desenvolvimento do Wigidrasil havia sido criada uma extensão, cujo propósito era armazenar os dados principais de um wrapper. Desta forma, utilizando um arquivo destes, um wrapper do mesmo tipo que o wrapper original e o mesmo motor utilizado neste, o segundo wrapper poderia se tornar uma réplica do primeiro.

A ideia era transportar um wrapper inteiro de uma maneira fácil utilizando um arquivo bem leve. A extensão era o WIG, derivada das primeiras letras de Wigidrasil, e seus arquivos pesavam meados de 1.1Mb. Os arquivos eram leves o suficiente para transporte, mas continham poucas informações e não eram capaz de recriar um wrapper do zero, necessitando sempre de um wrapper parecido já pronto.

Mesmo com o fim do Wigidrasil o suporte ao WIG foi mantido, mas por pouco tempo devido ao fato de ele não ter se popularizado. No entanto, quando o projeto Gamma Board foi recomeçado com a parceria com Paul Bertelink, a ideia de um arquivo leve que poderia recriar um wrapper voltou a ter ênfase.

Com isso, surgiu uma nova extensão: o WSI, ou *Windows Software Installer*. O WSI possibilitou criar não apenas um wrapper, mas um port inteiro a partir de uma

conexão com a internet. Com apenas um arquivo que consome entre 1Mb e 1.5Mb de memória, se tornou viável criar um port cujo peso poderia ultrapassar 200Mb.

Mas este método de “compactação” não foi criado sem nenhum planejamento em mente. A ideia, financiada por Paul Bertelink, autor do site Paul The Tall, era que os arquivos WSI gerados por ele e outros desenvolvedores fossem armazenados em um servidor, de onde poderiam ser baixados por meio do Porting Kit, e então instalados.

A criação desta extensão pode não apenas tornar a transferência de ports de uma pessoa para outra um processo mais simples, mas também pode mudar completamente a forma como o *porting* é visto nos dias de hoje. Pensando além, seria possível que um dia as empresas necessitassem apenas inserir um arquivo WSI dentro das mídias de seus jogos para que eles pudessem ser instalados em computadores Mac.

E para o caso de mídias digitais, as empresas necessitariam apenas submeter o WSI de seus jogos no servidor, e então qualquer um em um Mac com o Porting Kit seria capaz de instalar seus jogos sem nenhuma complicação. As aplicações voltadas ao porting passariam a ser apenas ferramentas para desenvolvedores, como sempre deveriam ter sido, e o Porting Kit seria a única aplicação necessária para se instalar qualquer programa de Windows no Mac. Mais detalhes sobre a forma de funcionamento do arquivo WSI serão vistos nos itens 6.1.14 e 6.2.6.

6.1.13 - Múltiplos Idiomas

O idioma da aplicação não é considerado menos importante. Todos os utilitários de wrappers possuem apenas o idioma inglês, o que é um padrão para programas puramente técnicos, mas que prejudica os usuários comuns que precisam/querem criar um wrapper.

O suporte a múltiplos idiomas está no Porting Center desde que ele era Appleverse Wizard, quando possuía os idiomas Português, Inglês (traduzido por mim), Espanhol (traduzido por Andréia Moreira) e Russo (Traduzido por Dmitry Porotnikov).

Ao se tornar Wigidrasil, depois da mudança de foco, boa parte dos termos mudaram, e como consequência as traduções precisaram ser refeitas do zero. Devido a isso, as traduções para espanhol e russo se perderam. No entanto, já como Porting Center, o programa recebeu os idiomas Francês (traduzido pelo usuário da Porting Team, DankoB), Italiano (traduzido por José Maria Simas de Miranda) e Alemão proveniente da Holanda (traduzido por Paul Bertelink).

A vantagem das múltiplas traduções é a facilidade de uso, a qual passa a ser fornecida a todos aqueles que conhecem uma das linguagens faladas. Uma prova de que isso é efetivo foi o aumento do número de downloads de países que falam um dos idiomas inseridos. Há planos para re-adicionar os idiomas Espanhol e Russo no futuro, assim como para conseguir tradutores para novos idiomas.

Vale mencionar que o Porting Kit também possui vários idiomas, sendo estes Português, Inglês (traduzido por mim), Holandês (traduzido por Paul Bertelink) e Alemão (traduzido por Sascha Lamprecht, usuária do Porting Kit).

6.1.14 - Manuseio de WSIs

Já estava planejado desde o começo que o objetivo era projetar um servidor que pudesse receber os arquivos WSIs e mandá-los a qualquer um que usasse o Porting Kit, e que o Porting Center deveria enviá-los para o servidor. Em determinado ponto, o Porting Center se tornou capaz de não apenas realizar uploads de ports, mas também de gerenciar todos os ports enviados por um usuário. Desta maneira foi possível realizar alterações ou pequenas mudanças em um port quando este apresentou problemas.

No começo, o Porting Center simplesmente criava uma pasta local com os arquivos necessários para que a aplicação ficasse disponível: os arquivos WSI, o arquivo WSII, o arquivo Genres.csv e o arquivo Icon.png, os quais serão explicados em seguida.

Primeiramente, o mais simples deles, é o Icon.png. Este arquivo é o ícone do programa, em resolução 128x128, utilizado pelo Porting Kit para que o jogo possa ser mais facilmente distinguido.

O segundo em matéria de simplicidade é o Genres.csv. Este arquivo CSV (*Comma Separated Values*: Valores Separados por Vírgulas) possui os gêneros do jogo (Exemplo: Ação, Aventura, RPG, etc), os quais são definidos durante o processo de exportação do mesmo. Ele ainda não foi amplamente utilizado, mas suas possibilidades são bem extensas.

O arquivo mais importante para identificação do port era o Description.wsii. WSII significa *WSI Identifier*, ou Identificador de WSI. Dentro dele existem diversos dados sobre o port: criador, descrição, trailer, data de criação, tipo (jogo completo, apenas wrapper ou VIP), ano de lançamento, empresa e os links afiliados. No entanto, como esse arquivo é criado via *NSArchiver* em Objective-C, somente um programa em Objective-C é capaz de lê-lo com *NSUnarchiver*, mas serviços online, por exemplo, não são capazes de lê-lo.

E por fim, os arquivos WSIs. Dentro da pasta existe um WSI para cada origem do arquivo, que pode ser de sites com os quais possuímos links afiliados ou de links diretos, dos quais se pode baixar o jogo diretamente.

A pasta era colocada manualmente com o FileZilla e o CyberDuck no diretório dos WSIs, no qual havia um arquivo PHP (listPorts) cuja finalidade era listar as pastas (e este era carregado pelo Porting Kit na época). Com esse método o sistema não suportaria uma carga de ports muito grande, e também não podia trabalhar com variáveis como o número de downloads e votações, os quais eram necessários.

Foi necessário utilizar um banco de dados, no caso o MySQL, para armazenar todos os campos necessários para gerir os ports. Além disso, foram criados seis arquivos PHP que acessavam o banco:

- **addPort:** Permite que um app seja adicionado à base;
- **download:** Realiza o download de um port incrementando seu contador;
- **newListPorts:** Lista os apps da base;
- **portRating:** Retorna a média de votos de um jogo em determinada placa de video;
- **rankingPorts:** Permite que os apps seja ordenados por popularidade, data de upload ou média dos votos;
- **ratePort:** Permite que o usuário vote no desempenho de um app.

Desses, apenas o primeiro foi utilizado pelo Porting Center. Além de adicionar o port à base também era necessário realizar o upload dos arquivos, o que foi uma tarefa difícil no começo. Existem várias opções para gerir FTP em Objective-C, mas a maioria usa Frameworks de terceiros, o que acabava fazendo o app consumir muita memória (sem falar no fato de que todos eram bem difíceis de configurar corretamente).

Por fim, foi usado o FTPManager, criado por Nico Kreipke e disponibilizado no GitHub. Uma vez configurado, ele serviu perfeitamente ao trabalho, e foi possível realizar o upload dos ports sem o uso de gerenciadores FTP, o que foi um grande avanço. Na figura 55 é possível ver o Porting Center com esse recurso implementado.

Figura 55: Layout final do Porting Center

Somando isso ao gerenciamento de fontes (locais onde o jogo foi obtido), o Porting Center passou a tomar controle total dos processos de adição e alteração de ports da base de um determinado usuário, deixando de lado apenas a remoção, a qual por questões de segurança achamos melhor manter de maneira manual.

Para se enviar um port para o servidor é necessário possuir uma conta. Para isso, é criada uma conta FTP para cada desenvolvedor. Para estender as possibilidades e facilitar os processos, as pastas acessíveis por essas contas encontram-se disponíveis publicamente (*public_html*) dentro do diretório "*wsis"*.

Dentro desta pasta existe um diretório para cada usuário, tendo esse o nome do seu respectivo usuário e sendo o diretório home de seu usuário FTP, sendo esse o único lugar em que é possível realizar operações. Desta maneira, é possível saber quem enviou cada port e limitar o poder de alteração do usuário aos seus próprios ports.

No final de 2015, a base do Porting Kit teve um grande mudança para facilitar o acesso à informações online. Os arquivos WSII e CSV foram removidos, e seus dados passados para a base MySQL. Isso deu origem a mais três arquivos PHP para acesso a base:

- **portDescription:** Utilizado pelos dois programas e pelo site para obter as informações da base de um port específico.
- **setPortGenres:** Usado apenas pelo Porting Center, com o objetivo de definir quais os gêneros de um port específico.
- **setPortSources:** Também usado pelo Porting Center, que serve para definir quais as fontes e respectivos links de um port.

Essa alteração permitiu a criação de uma página de busca no site do Porting Kit (figura 56), com direito a filtros, o que facilitava ainda mais os usuários a obterem informações sobre os jogos que desejam instalar, sem precisar instalar o aplicativo desejado.

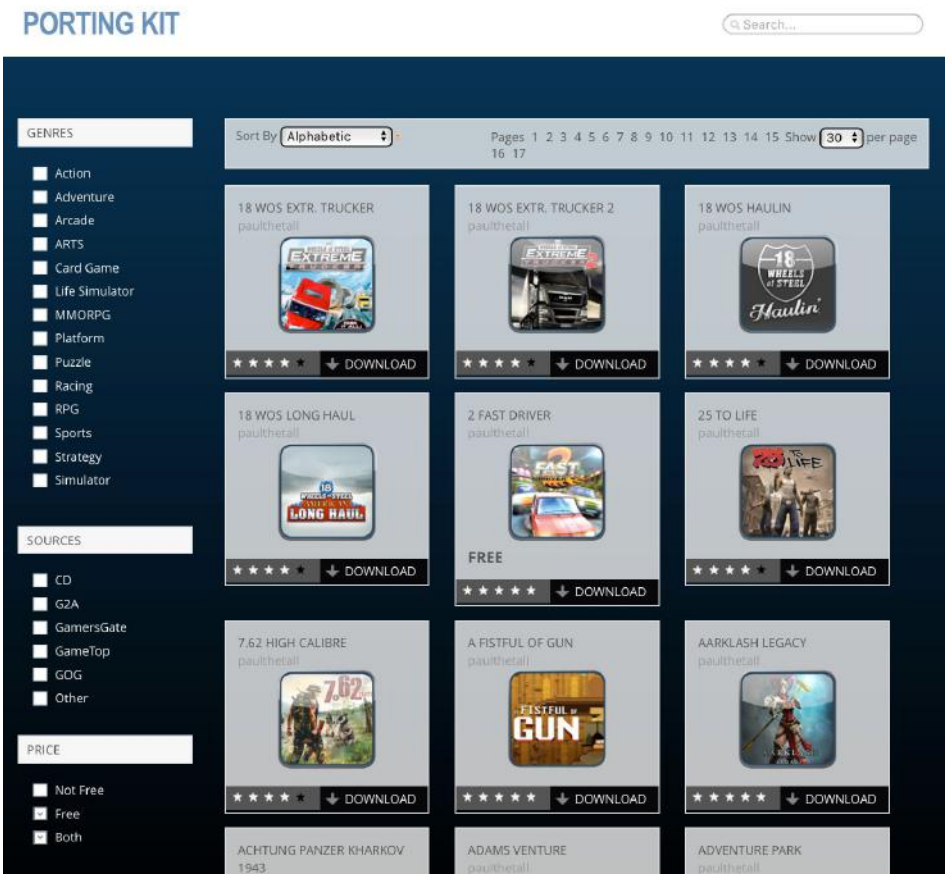

Figura 56: Página de busca do Porting Kit

Além da página de busca, ao selecionar um port era possível ver a mesma descrição disponibilizada no app, junto ao vídeo de trailer, os gêneros e as fontes, como mostrado na figura 57. Isso permitia que o usuário verificasse as informações de um port pelo site.

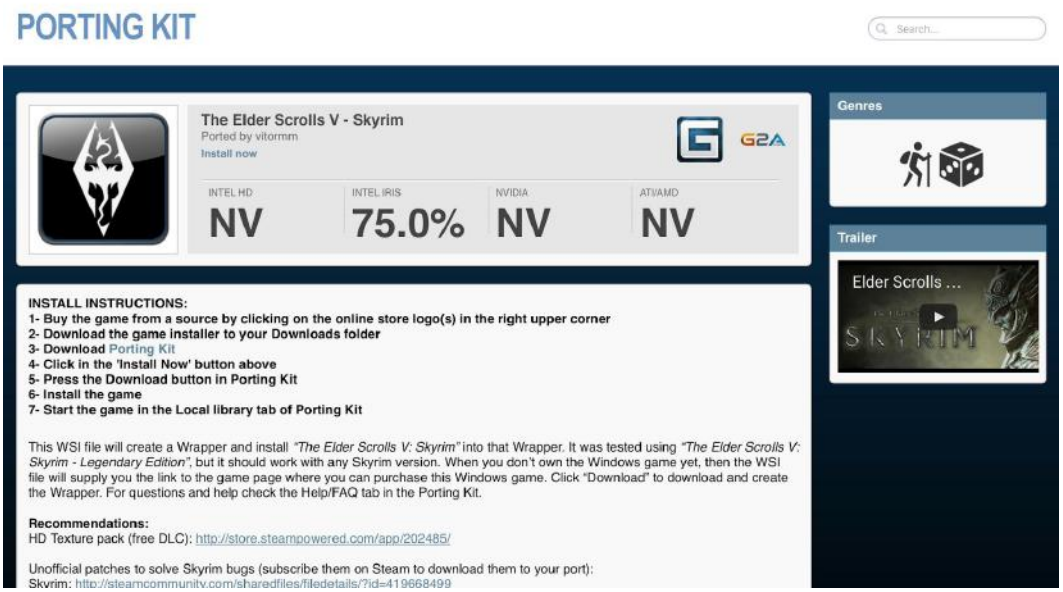

Figura 57: Página do jogo "The Elder Scrolls V: Skyrim"

Apesar disso, o layout não era muito agradável, e não era compatível com o resto do layout do website. Para resolver o problema, durante uma "repaginada" que o website passou no início de 2017, as páginas de busca (figura 58) e de resultado (figura 59) também foram mudadas.

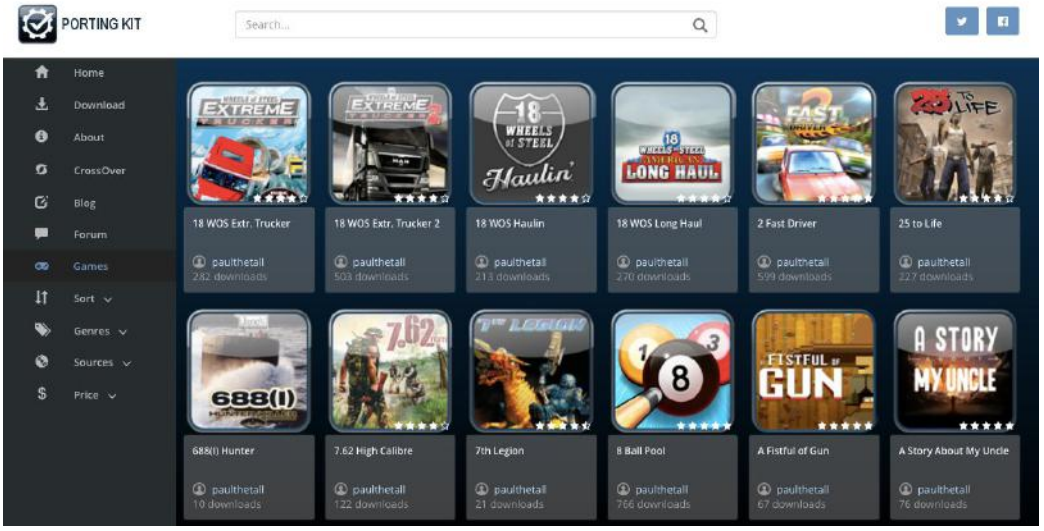

Figura 58: Nova página de busca do Porting Kit

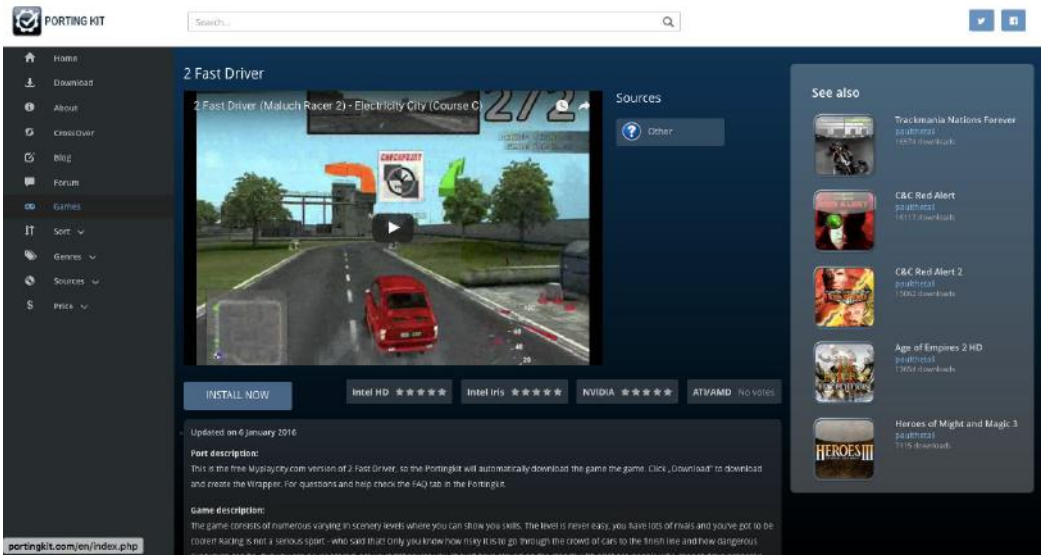

Figura 59: Nova página do jogo "2 Fast Driver"

Pouco antes do Porting Center se combinar ao Porting Kit, a forma de fazer upload dos arquivos mudou, assim como a forma de obter informações do servidor. Os arquivos passaram a ser enviados para uma pasta com seu ID em vez de uma pasta com seu nome, evitando possíveis problemas que isso poderia causar no futuro.

Originalmente, as informações eram retornadas de forma rudimentar. Cada linha da saída correspondia a uma aplicação, e os dados de cada aplicação eram separados pela *string* " || ", e se esperava por cada dado em uma posição específica. Como isso causava alguns problemas caso houvesse falha durante o recebimento de listagens de apps ou ao obter a descrição de app, esse método foi substituído por um JSON, que passou a poder carregar muito mais informações em segurança, já que o método anterior era muito instável em comparação. Para isso foram necessários mais 6 arquivos PHP, substituindo quase que completamente a estrutura original:

- **addApplication:** Permite que um port seja adicionado à base;
- **appDescription:** Utilizado pelos Porting Kit para obter as informações sobre um app específico.
- **appRating:** Retorna a média de votos de um app em todas as placas de video;
- **listApps:** Lista os ports da base;

- **ranking:** Permite que os apps seja ordenados por popularidade, data de upload ou média dos votos;
- **rateApp:** Permite que o usuário vote no desempenho de um app.

Diferentemente do Porting Center, o Porting Kit não possui uma interface exclusivamente para o gerenciamento de aplicativos presentes na base de dados. Em vez disso, a Biblioteca, que será melhor explicada no capítulo 3, passa a contar com opções especiais para o usuário que desenvolveu cada aplicativo, como *“Editar aplicativo”*, *“Editar port”* , *“Editar link de fonte”*, *“Adicionar fonte”* e *“Remover fonte”*.

Nem todos os recursos já foram re-implementados no Porting Kit, mas o Porting Center já se encontra fora de operação.

6.2 - PORTING KIT

Desde o início do projeto havia a ideia de que era necessário encontrar uma maneira de se facilitar a instalação dos programas utilizando o Wineskin. Bem no começo, eu acreditava que seria possível criar um programa que pudesse portar um aplicativo sem necessitar de auxílio humano, mas isso se mostrou uma tarefa complexa demais.

Desde a criação do arquivo WIG até os atuais WSI (apresentados no item 6.1.12) havia o conceito de que era necessário pelo menos um usuário de nível suficientemente alto para portar o programa uma vez e, desta forma, outros poderiam usufruir de sua descoberta. Isso já acontecia quando os ports eram passados via internet, porém era preciso ter algo mais estável, mais confiável e mais rápido.

O Appleverse Wizard já possuía essa ideia, no entanto ele apresentava apenas instruções legíveis para um ser humano, o que fazia dele um tipo de manual de instruções. Era útil, mas não facilitava o trabalho de usuários inexperientes. Os arquivos WIG já possuíam seus dados legíveis para uma máquina, porém não carregavam muitas informações consigo, tendo suporte a apenas um executável e sem possuir preferências de tela, por exemplo.

Quando os arquivos WSI surgiram a ideia começou a tomar forma, e depois que a parceria com PaulTheTall foi firmada surgiu o conceito do Projeto Porting.

O Porting Kit é o único programa do seu gênero que, utilizando os arquivos WSI gerados pelo Porting Center (ou WSI2 gerados pelo Porting Kit), pode criar uma cópia exata de um port em outro computador. Desta forma, é muito mais simples compartilhar um port com outra pessoa (ainda mais considerando que um port baixado da internet costuma conter entre 200Mb e 600Mb, ainda vazio, e também é necessário desativar a proteção contra softwares não assinados para usá-lo, o que pode deixar o computador vulnerável).

Obviamente, a instalação de arquivos WSI é um de seus recursos principais. Nas próximas seções, os recursos do Porting Kit serão explicados com clareza.

Demorou muito tempo para se chegar em um design definitivo, afinal deveria ser o mais claro possível para o entendimento de qualquer usuário. Inicialmente foi projetado o layout, mas não era visualmente agradável, como pode ser observado na figura 60.

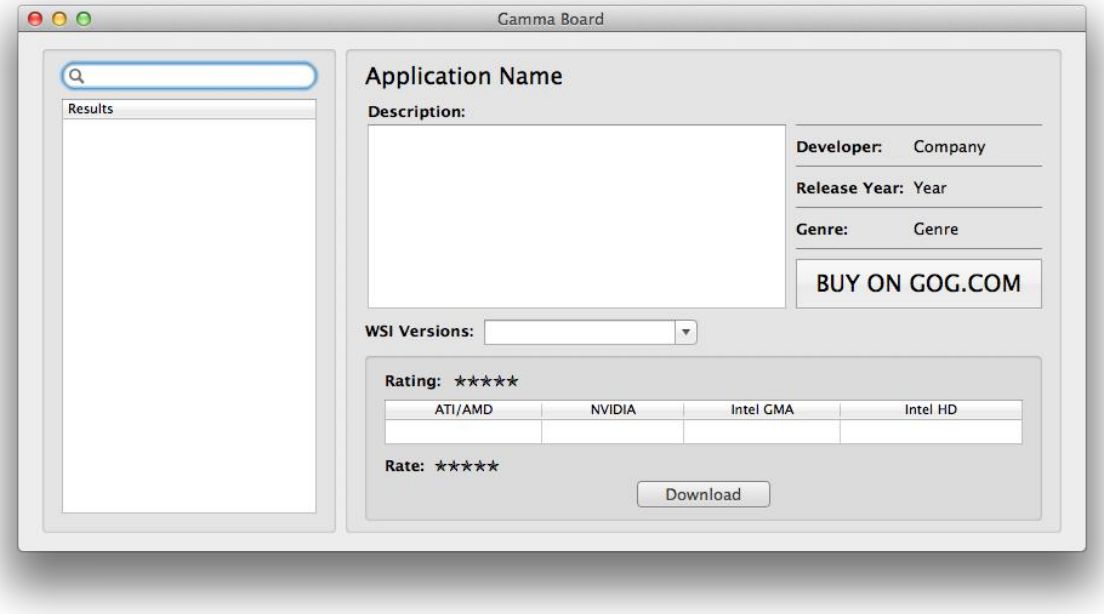

Figura 60: Primeiro layout do Porting Kit; antes a Biblioteca era a totalidade do app

Ficou claro que mudanças eram necessárias. A área de votação e de Download foram unidas, pois deveriam remeter ao mesmo port, mas isso se tornava

redundante visto que as diferenças entre diferentes WSI de uma mesma aplicação eram mínimas, tornando seus resultados quase idênticos.

Apesar disso, era suficiente para se realizar os testes, e por muito tempo se manteve assim, mas sem ter acesso a um servidor e sem ter sido lançado a público. Em uma conversa com Paul, o layout foi totalmente remodelado, conforme apresentado no sketch da figura 61.

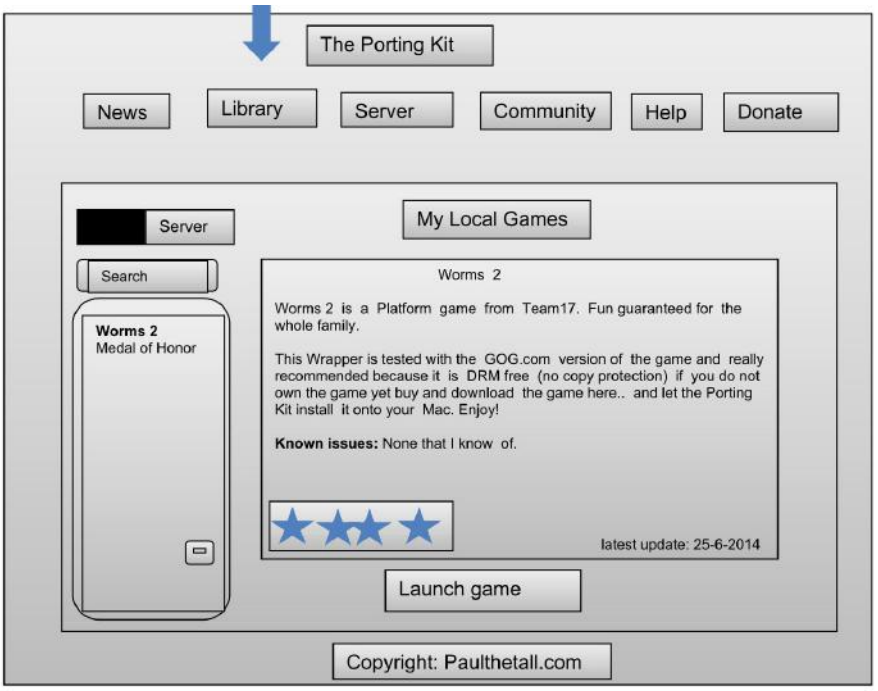

Figura 61: Sketch original da aba da Biblioteca (aba Local)

Baseado neste sketch, que era bem mais amigável, foi criado o novo visual do Porting Kit, o qual está presente na figura 62.

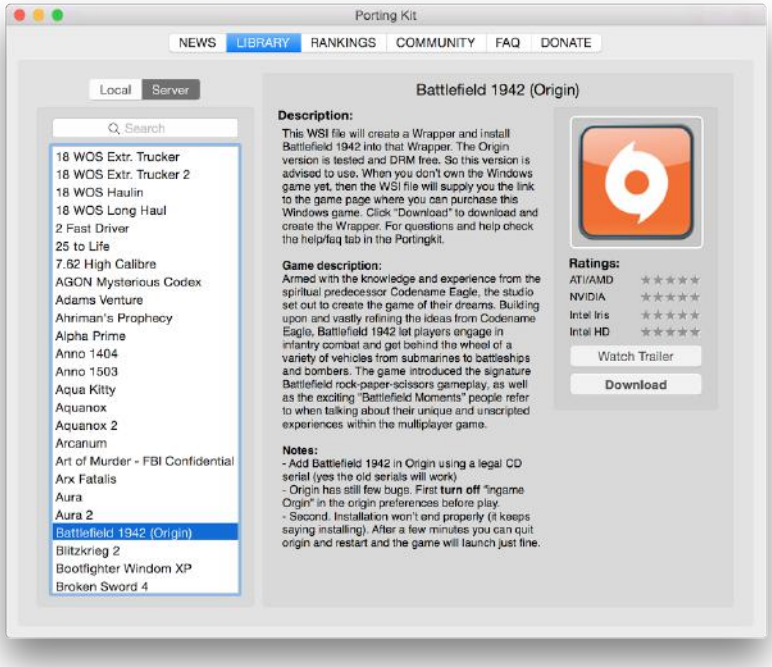

Figura 62: Layout baseado no sketch original

Parte dos recursos foi planejada logo de início, e outros surgiram no decorrer do desenvolvimento. Em fevereiro de 2015, Paul contratou um designer, Subhan Ahmed, para nos ajudar a realizar aprimoramentos no visual do programa, o qual propôs o layout apresentado na figura 63, amplamente inspirado no programa Steam.

Figura 63: Layout originalmente proposto por Ahmed para o Porting Kit

Ele imaginava que para cada jogo selecionado o plano de fundo mudaria, assim como ocorre ao selecionar jogos pelo *Steam*. No entanto, eu e Paul concordamos que a aplicação deveria possuir um background único, e que certas coisas estavam longe do ideal. A tela então foi redesenhada várias vezes, até chegar ao seu estado final presente na figura 64, com focos nos tons azul e preto, como no ícone da aplicação.

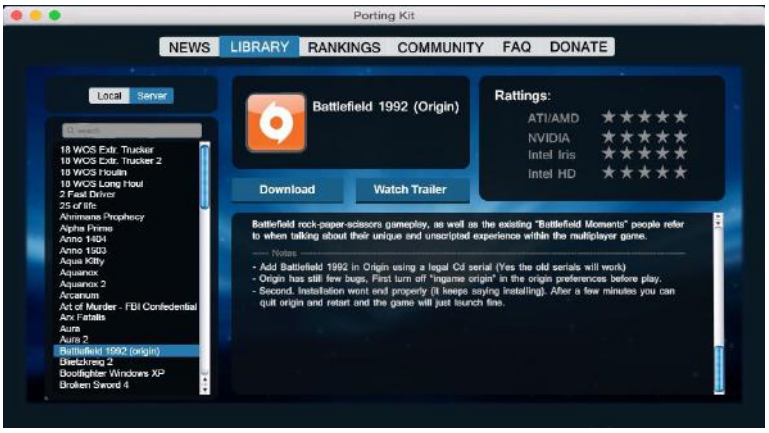

Figura 64: Layout final de Ahmed para o Porting Kit

Com um design definido, a última etapa foi transformá-lo em aplicação. Certas partes foram bem mais difíceis que outras, como a borda do background que precisava mudar de tamanho sem distorções conforme a janela do programa era redimensionada, mas apesar das dificuldades o resultado foi muito satisfatório (figura 65).

Figura 65: Layout do Porting Kit adaptado para janelas redimensionáveis

Neste ponto, o Porting Kit se tornou capaz de adicionar aplicações nativas à sua biblioteca local. Isso foi feito para aumentar a arrecadação do aplicativo e também para centraliza-lo como o gerenciador de programas de computador, promovendo o seu uso.

A janela se manteve desta forma por muito tempo. No entanto, a versão 2.0.0 sofreu uma grande alteração, se inspirando menos no Steam e mais em programas como Origin e GOG Galaxy. A lista foi substituída por uma *Collection View*, que é uma visualização em forma de grade que permite aos itens mudarem de posição conforme a janela for redimensionada.

O primeiro problema foi o plano de fundo, o qual passou a incluir um Dock como o presente em antigas versões do Mac OS X. O Dock não ficava alinhado com os itens da coleção quando a área ocupada pelos ícones era menor do que o tamanho da *view* da coleção, isso por que a imagem começava no canto inferior esquerdo da área de *scroll*. Para contornar isso, o Dock passou a ser recortado na horizontal de acordo com a posição dos itens em dois pedaços que se invertiam, permitindo que o Dock se ajustasse de acordo com a posição dos itens. Esses dois pedaços se repetem alternadamente para baixo até atingir a altura da *view*.

O sistema de ajuste de tela usado até aqui era o Auto-Layout, fornecido pela Apple desde o OS X 10.7, o que impedia que janela fosse redimensionada na versão *Legacy* do programa. No entanto, já pensando no redimensionamento dos ícones que viria no futuro, o Auto-Layout foi substituído pelo sistema de *constraints*, o qual era usado pela Apple anteriormente e era mais adequado para as necessidades do programa.

Posteriormente, adicionou-se a barra para alterar o tamanho dos ícones. Fazê-los mudar de tamanho foi difícil, mas o mais complicado foi fazer o Dock acompanhar essa mudança de tamanho. No fim das contas o resultado foi tão bom quanto o esperado.

As mudanças podem ser vistas em sua totalidade na figura 66.

Figura 66: Layout da biblioteca do Porting Kit na versão 2.1.0

Como consequência, a descrição de cada aplicação foi movida para uma tela diferente (figura 67). Ao clicar no ícone uma segunda tela com nome, ID, trailer, média dos votos, fontes e descrição era exibida, além de ter botões para votar, instalar/executar e um botão de 'Mais', o qual exibe um menu específico para cada aplicação.

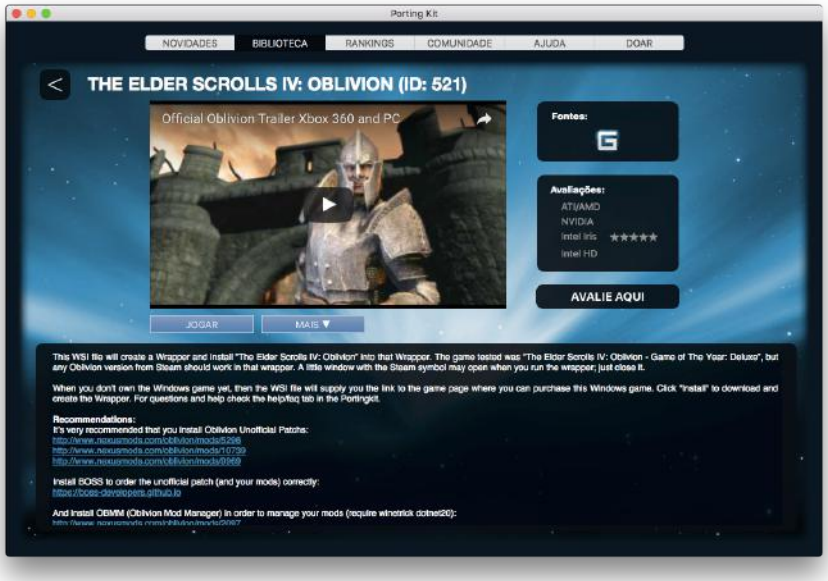

Figura 67: Descrição de uma aplicação

Mostrar as fontes, as quais eram botões para adquirir a aplicação antes da instalação do port, foi algo também adicionado ao antigo layout, o qual ainda podia ser usado caso o usuário desejasse; bastava ir às Preferências do programa e habilitar o uso da *Biblioteca Clássica*. Atualmente é possível inverter para o antigo layout por meio de um botão no canto superior direito.

Também passaram a existir opções extras, as quais estão disponíveis no clique direito do ícone, no clique direito do item no antigo layout e ao pressionar o botão 'Mais'. Dependendo do port, os opções extras variam.

Figura 68: Menu de clique direito para um port local e para um port no servidor na versão 2.1.0

Um port no servidor teria as opções de 'Baixar WSI' e de 'Instruções', as quais poderiam respectivamente permitir ao usuário baixar o WSI para realizar a instalação depois, ou de assistir as instruções para realizar a instalação de um port. Caso fosse uma aplicação nativa, a primeira opção não estaria visível. A opção de 'Instalar' aparece caso seja um clique direito no ícone ou no item de um port no servidor.

Já um programa no disco local teria as alternativas 'Forçar Encerrar', 'Remover', 'Propriedades', 'Mostrar Conteúdo do Pacote' e as específicas de ports Wineskin. O primeiro funcionava de forma similar ao botão *'Kill Wineskin Processes'* do Wineskin, mas finalizando apenas os processos de um port específico; só era selecionável se fosse um port Wineskin.

O botão de Remover oferecia ao usuário a possibilidade de remover o item da lista ou do disco. O botão de Propriedades mostrava uma janela que permitia ao

usuário realizar alterações de um port já instalado. O botão ‘Mostrar Conteúdo do Pacote’ funcionava como a opção de mesmo nome que aparece ao dar um clique direito em uma aplicação por meio do Finder, navegador de arquivos do Mac OS X, isso até o mesmo ser substituído pelo botão ‘Abrir drive C:’ (um para cada drive associado ao port), o qual abria o diretório drive_c que cada port, que é o que possui maior importância para o usuário comum, como foi observado rapidamente pelas necessidades dos usuários. Também havia o botão 'Instalar Mod/Patch’, criado pensando em jogos, que permite copiar arquivos para dentro de um port, extrair um arquivo compactado dentro de um port ou mesmo executar um .exe específico com o port, para realizar uma instalação por exemplo.

Os botões específicos do Wineskin eram para usuários mais avançados, e só eram visíveis se o port selecionado fosse um wrapper Wineskin. Primeiramente, havia o botão de Debug, que executava o equivalente a um *Test Run* do Wineskin, retornando logs de uso ao fim de uma execução.

Em seguida, estavam disponíveis os botões dentro do submenu 'Ferramentas do Wineskin’. A primeira opção, separada das demais, era ‘Executar Wineskin’, a qual executava o programa Wineskin contido dentro do wrapper. Logo após, foram colocadas quatro opções: ‘Gerenciador de Tarefas’, ‘Prompt de Comando’, ‘Editor de Registro’ e ‘Configuração do Wine’; todas chamavam comandos internos do Wine, que chamam aplicações de mesmo nome que estão contidas em cada wrapper, das quais apenas a última não existe no Windows tradicional.

A 'Configuração do Wine’ (ou *winecfg*, como é mais conhecida) permite ao usuário configurar o Wine do wrapper, o que permite realizar várias alterações como alterar os discos do mesmo, manusear a substituição de bibliotecas DLL, entre várias outras tarefas.

Outra mudança que a nova janela (figura 66) ofereceu é a remoção da janela de progresso. Desde suas primeiras versões o Porting Kit mostrava o progresso de suas ações em uma janela a parte, a qual ficava sobre as demais. Isso se mostrou ser um incômodo para o usuário, mas não havia muito o que se fazer.

Quando o Auto-Layout foi removido se tornou mais fácil criar uma área da janela visível apenas em certos momentos, e com isso foi adicionada uma área de progresso, só visível durante operações.

Figura 69: Nova janela com a área de progresso

Apesar disso, em versões posteriores, a criação de novos ports voltou a ser feita em uma janela a parte, que no caso era uma janela inspirada pelos *packages*, instaladores de programas para macOS, utilizados por programas como *TeamViewer* e *VirtualBox*, como como pode ser visto na figura 70.

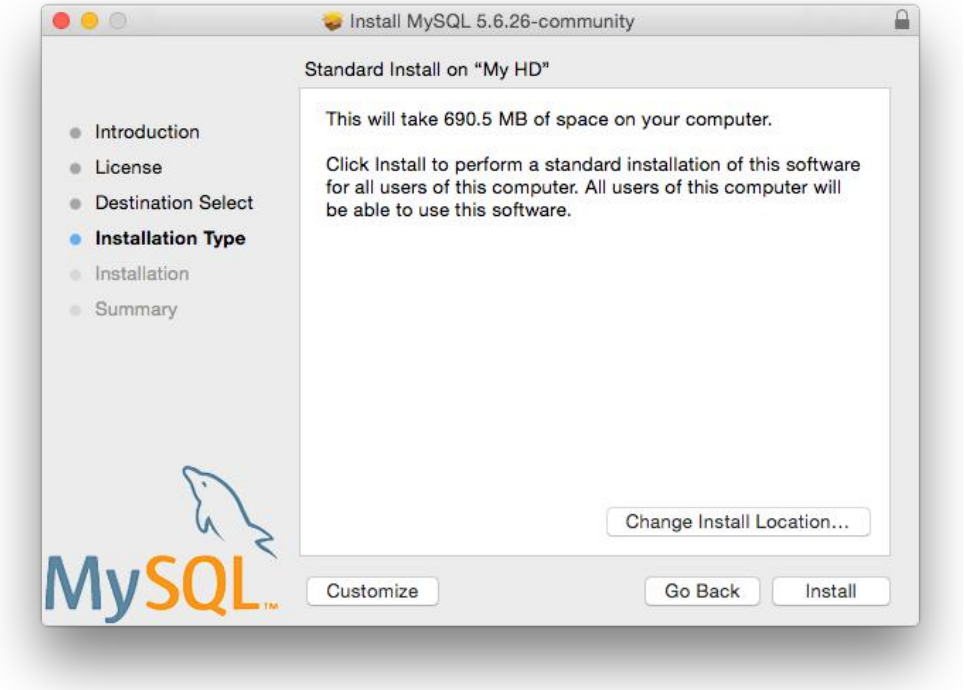

Figura 70: Package de instalação do MySQL Community

Essa decisão foi tomada para que fosse mais simples para o usuário saber em que etapa a instalação se encontrava, e também permitir um controle maior sobre a instalação do port, como: onde ele seria instalado, quais os arquivos

necessários para sua instalação, entre outras coisas. Um exemplo dessa nova janela em atividade pode ser observado na figura 71.

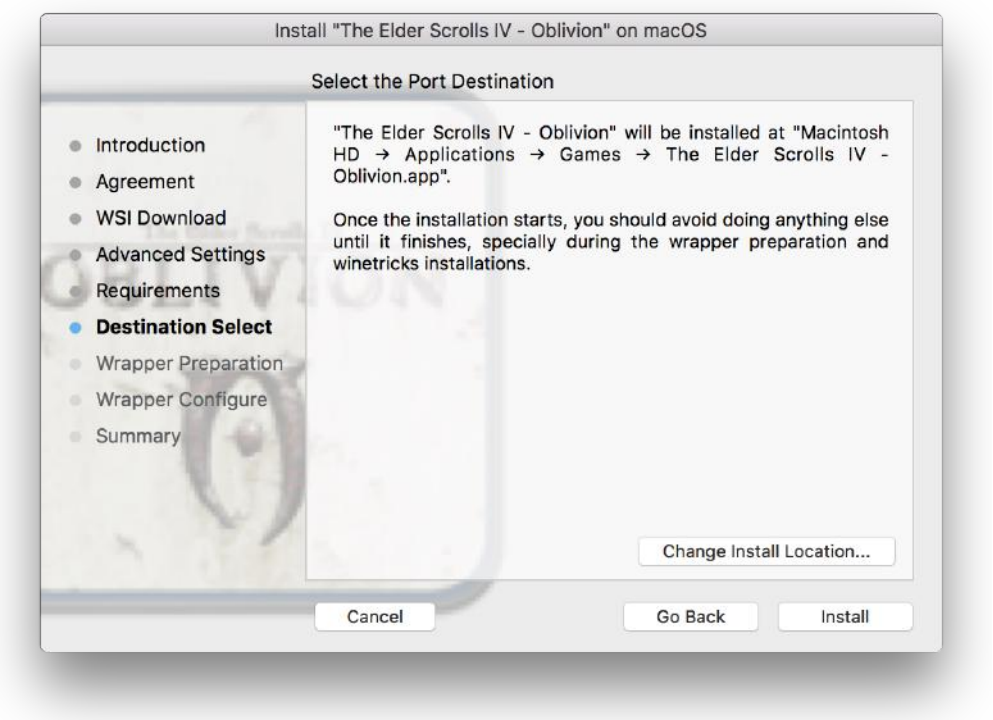

Figura 71: Janela de instalação the "The Elder Scrolls IV - Oblivion" no Porting Kit

Uma lista com todas as mudanças já feitas no Porting Kit pode ser encontrada no anexo 2.

6.2.1 - Novidades

A aba Novidades é um WebView com visão direta para o site paulthetall.com, o qual é o divulgador e financiador do projeto. Esta aba permite que as pessoas sejam informadas das atualizações do programa com mais clareza e rapidez. Não há muito o que se aprofundar neste tópico. Sabíamos que uma aba assim seria útil para facilitar a comunicação com os usuários.

A única mudança que essa aba recebeu no decorrer das versões foi que antes iniciava no topo da página paulthetall.com, e atualmente ela começa no topo da primeira postagem do site. Durante um determinado período de tempo um blog

havia sido criado dentro do domínio portingkit.com para substituir paulthetall.com, mas isso não durou muito tempo.

6.2.2 - Biblioteca

A aba Biblioteca foi inspirada em famosos clientes de DRM de jogos, os quais estão presentes em todos os sistemas, apesar de possuírem uma gama de jogos muito maior para o Windows do que para os outros sistemas. Sua aba se divide em duas partes: A aba *Local* e a aba do *Servidor*.

A aba Local lista os ports contidos na pasta de Aplicativos do usuário (ou em outra pasta de sua preferência; isso pode ser trocado na janela Preferências), e pode ser usada para iniciar os jogos diretamente pelo programa, reler a descrição do aplicativo se necessário, *debugar* a aplicação (o que pode ser feito pressionando Jogar enquanto se segura a tecla Shift), remover um port do computador, executar o utilitário padrão do port, mostrar os conteúdos de um wrapper ou buscar um port em meio aos outros por meio da barra de pesquisa. Além disso, é possível adicionar outras aplicações, inclusive *não-ports*, para aumentar a comodidade do usuário.

A aba Servidor lista os jogos cujos ports estão disponíveis por meio de arquivos WSI no servidor. Estes estão disponibilizados onde se hospeda a página portingkit.com, e variam de acordo com a origem do instalador que é usado na instalação. Na maioria dos casos os instaladores vinham de cópias digitais dos sites GOG.com e GamersGate, ou eram instalados via Steam, mas o Porting Kit aceitava instaladores de virtualmente qualquer fonte, inclusive de outros clients de DRM como a Origin.

Nesta mesma aba também é possível utilizar e visualizar o sistema de votação, com o qual é possível saber e informar o desempenho daquela aplicação específica na sua placa de vídeo. O usuário só é capaz de votar na placa de vídeo correspondente ao seu computador, o que é verificado pelo programa por meio do seguinte comando de terminal:

system_profiler SPDisplaysDataType

A saída do comando era obtida pelo programa e trabalhada. Por meio desta era possível obter a variável *Chipset Model*, a qual informa se a placa de video presente é Intel HD, Intel Iris, ATI/AMD ou NVIDIA, o que *filtrava* o voto do usuário para a sua placa específica.

O voto era adicionado à nossa base, mas antes era feita uma autenticação via Facebook, o que foi feito para impedir votos duplos e spam, garantindo maior veracidade nos valores. Pensamos em outras formas de autenticação, no entanto essa foi a que pareceu mais eficaz e que iria requerer menos trabalho por parte do usuário. A janela de votação pode ser vista na figura 72.

Figura 72: Janela de votação

Junto com os resultados da votação, o usuário também podia visualizar uma descrição do aplicativo fornecida por quem o adicionou ao servidor. Essa descrição pode conter auxílio para bugs conhecidos, assim como informações sobre o jogo e dicas para melhorar o seu desempenho.

Toda vez que o usuário baixa um port por meio do Porting Kit, um programa PHP realiza um incremento na variável do número de downloads daquele port, o que nos permite visualizar quais os ports mais populares, algo que será mostrado no item 6.2.3.

6.2.2.1 - Propriedades

A janela Propriedades (figuras 73 e 74) pode ser acessada por clique direito ou pelo uso do botão 'Mais'. Ela foi criada inicialmente por dois motivos: economizar o tempo dos usuários que desejassem mudar o ícone, nome ou versão de um app, e permitir a eles mudar o caminho de execução de um wrapper da mesma maneira amigável que o Porting Center oferecia.

Durante um tempo o Porting Kit oferecia durante a criação de um wrapper se o usuário gostaria de mudar a fonte padrão do mesmo para Helvetica Neue, a qual era a fonte padrão do Mac OS X até a versão 10.10. Isso foi substituído por uma caixa de seleção na janela de propriedades. Também foi adicionada uma caixa para habilitar/desabilitar o Wine System Tray (Capítulo 6.2.6.8).

Posteriormente (versão 1.9.125) foi adicionada uma segunda aba a essa janela, aonde o usuário poderia alterar, ou inserir a sua própria, descrição do aplicativo. Isso foi feito a pedido dos usuários, principalmente para aqueles que já possuíam wrappers Wineskin em seus computadores quando o Porting Kit foi instalado.

Figura 73: Aba para edição da descrição

E por fim, foi adicionada uma caixa de seleção que permite ao Porting Kit inverter o funcionamento das teclas F1-F12 durante a execução de um port por meio deste. Isso foi feito por um simples motivo: algumas aplicações usam essas teclas, e

por padrão você precisa usar a combinação Fn mais a tecla para executar a função normal no OS X. Caso contrário estará executando uma função do sistema, como controlar a luminosidade e o som.

As primeiras duas caixas de seleção podem ser usadas somente em ports Wineskin, assim como o campo de executar. As outras opções podem ser usadas para qualquer tipo de aplicação.

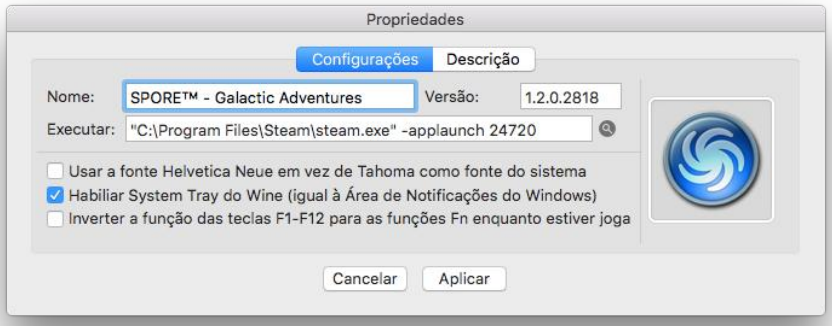

Figura 74: Aba de Configurações

6.2.3 - Rankings

Esta aba foi criada baseando-se no conceito de TOP 10. Em 3 colunas, é possível ver os 25 ports mais recentes, mais baixados e com melhores notas. Essa ideia esteve presente desde que o conjunto de sketchs foi feito, como pode ser observado na figura 75, e tem como objetivo ajudar as pessoas a conhecer melhor as opções de jogos que podem instalar.

Utilizando PHP, cada uma das listas acessa individualmente o servidor, obtendo sua respectiva listagem. Por meio de uma simples condicional, o PHP sabe de que maneira deve ordenar a lista que irá enviar.

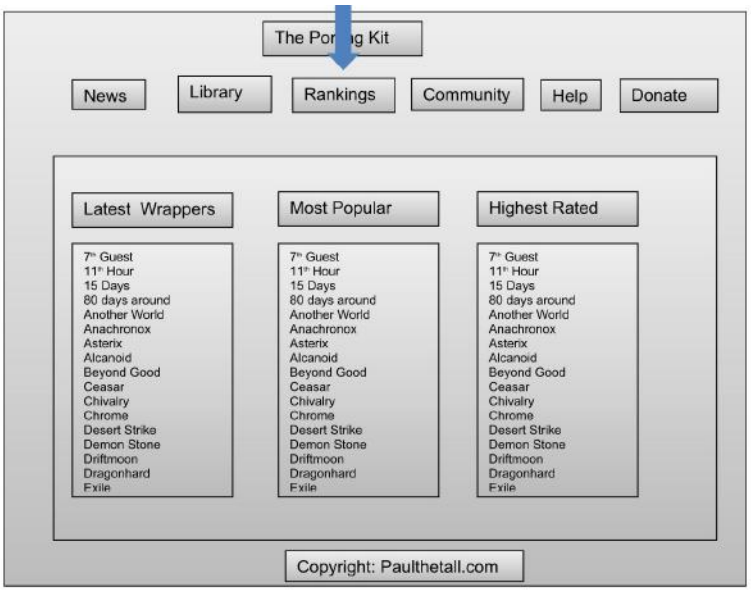

Figura 75: Sketch original da aba de Rankings

Caso seja a tabela de tipo *latest*, ele irá ordenar os ports pela data de upload. Caso seja do tipo *popular*, irá ordenar os ports pelos que tiveram o maior número de downloads. E por fim, se for *highrated*, os ordenará pela média de seus votos, contando como zero se não houve voto algum.

6.2.4 - Comunidade

Apesar de essa idéia estar presente desde os primórdios do app, muita coisa mudou na sua maneira de ser, como pode-se observar no sketch original na figura 76.

Originalmente, cada aplicativo disponível para download deveria ter uma seção no chat, pelo qual as pessoas poderiam conversar livremente sobre assuntos relacionados ao referido jogo. Além disso, também foi planejado um fórum, onde as pessoas poderiam postar caso seus problemas fossem mais recorrentes ou não tivessem sido resolvidos tirando dúvidas pelo chat.

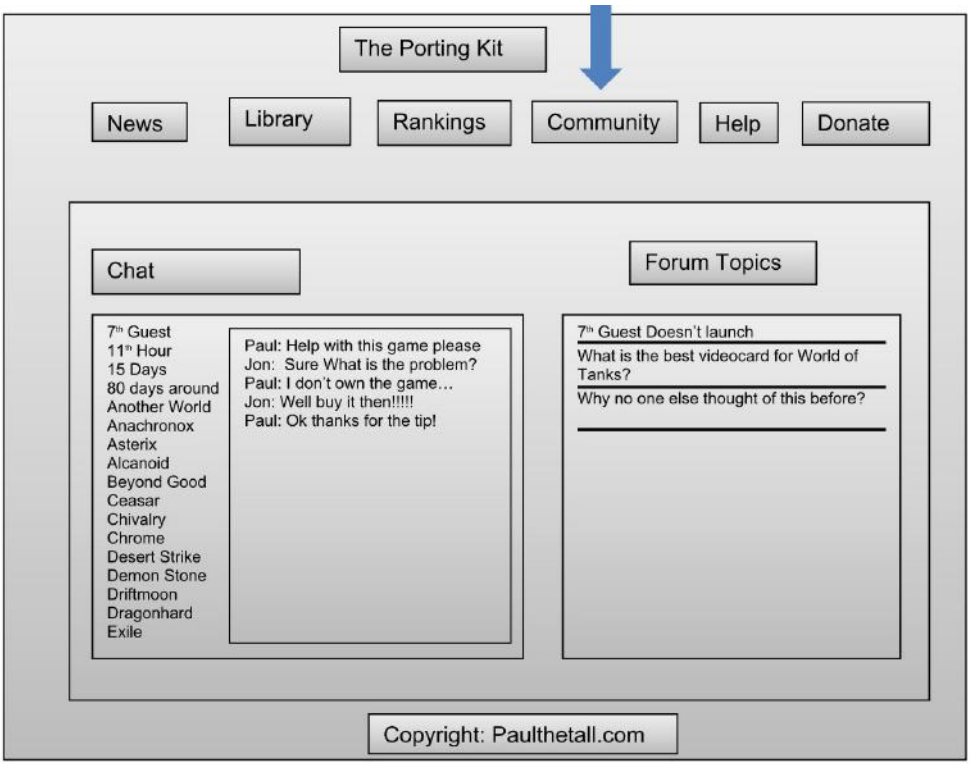

Figura 76: Sketch original da aba de Comunidade

Isso trazia implicações, como a necessidade de alguém monitorando o chat para evitar seu mal uso, e seria necessário a implementação de métodos para evitar spam, ou poderíamos ter um grande problema. O tempo se passou, e esse recurso acabou sendo um dos últimos implantados; justamente quando passou a ser bem necessário.

Muitas pessoas começaram a enviar mensagens pedindo pelo port de certos jogos, no entanto era difícil avisar a todos os que pediram quando um port estivesse pronto, e da mesma maneira seria interessante se pudesse informar a todas as pessoas que uma determinada aplicação já estava na nossa lista de pedidos. Além disso, era preciso haver um lugar em que os usuários pudessem interagir com liberdade.

No fim, a aba de Comunidade se tornou um WebView, acessando um fórum disponível no domínio do Porting Kit. O fórum foi criado utilizando o SMF (Simple Machine Forum), uma tecnologia open-source para criação de fóruns, a qual foi a alternativa gratuita mais interessante e melhor elaborada que encontrei na época.

Figura 77: Aba de Comunidade em sua versão final

Por meio deste fórum (figura 77), os usuários do programa conseguiam se comunicar, relatando seus problemas, dando sugestões, entre outras coisas. A página de comunidade tinha pacotes adicionais para dar aos usuários mais opções de avatares, e outros para reduzir o problema de spam.

Qualquer usuário podia postar ou comentar na Comunidade, mesmo que não estivesse registrado. Isso deixava os usuários a vontade, e permitia que eles usufruíssem do Porting Kit sem compartilhar quaisquer informações pessoais.

Um problema que começou a afetar o fórum de forma intensa após certo tempo foi o excesso de usuários e posts spam, os quais conseguiam passar facilmente pelo sistema padrão de Captcha do SMF. Os spams eram reportados pelos usuários e admins, e então apagados pelos admins quando havia tempo. Depois de um longo tempo, quando houve um aumento brusco na quantidade de spam, esse sistema teve de ser substituído pelo ReCaptcha do Google, o que resolveu o problema dos spams de maneira definitiva e imediata.

6.2.5 - Ajuda e Doar

A aba de Ajuda no início tinha um o nome de FAQ, possuindo internamente abas de Instrução, Ajuda e Contato, no entanto percebeu-se que não fazia sentido colocar a aba de contato dentro da aba de FAQ, e achou-se melhor inverter os nomes de FAQ e Ajuda.

A aba interna de Introdução nunca chegou a ser preenchida, sempre deixada para depois. A ideia era inserir instruções básicas do funcionamento do Porting Kit, mas como o funcionamento do programa e a interação com os usuário foi priorizado, ela, assim como todas as abas de Ajuda, tiveram pouca atenção.

A aba de FAQ (Frequently Asked Questions) possuía perguntas que foram feitas com muita frequência pelos usuários via e-mail, ou que foram consideradas curiosidades comuns. No caso da figura 78, as perguntas 1, 2 e 5 foram feitas com muita frequência, e as perguntas 3 e 4 foram colocadas mais a critério de curiosidade dos usuários. As perguntas foram sendo alteradas com o tempo de acordo com as necessidades e as atualizações do programa.

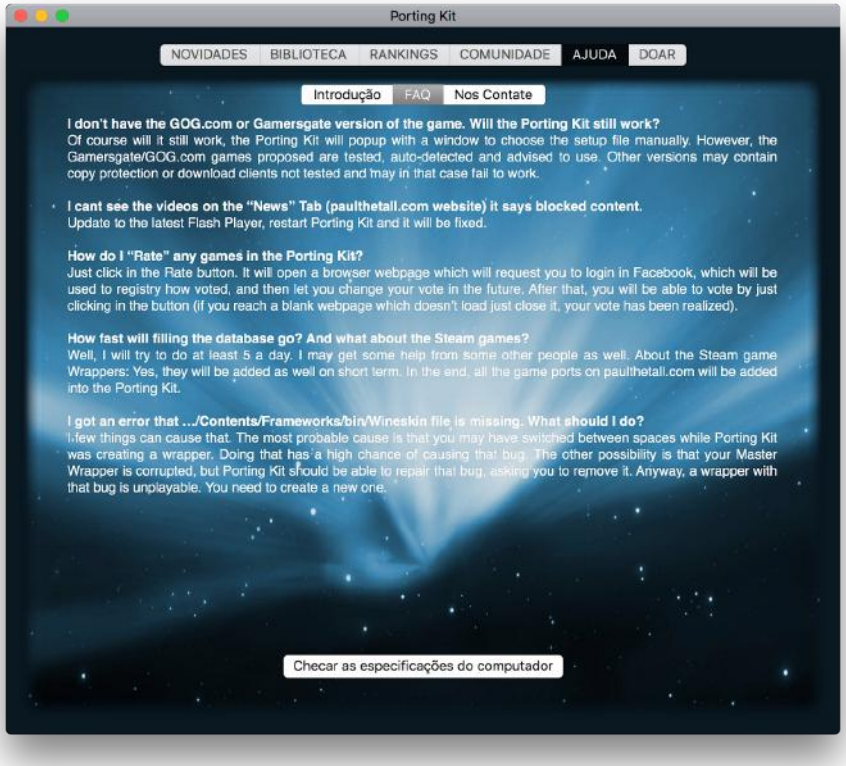

Figura 78: Aba de FAQ, com as perguntas consideradas pertinentes

Outro problema muito comum e constantemente necessário era o usuário desconhecer as especificações do próprio computador. Como o funcionamento dos ports pode variar dependendo da máquina, essa era uma informação muito necessária, e com alguns usuário isso consumia muito tempo.

Em um primeiro momento, foi inserido o botão mostrado na parte inferior da imagem acima, o qual permitia ao usuário conferir as suas especificações em um diálogo, como mostrado na figura 79. Mesmo assim muitos usuários se complicavam no processo de ter de clicar no botão e mandar as informações, fora o fato de que isso era quase sempre necessário. Esse problema foi resolvido colocando as especificações do usuário dentro do corpo de todo e-mail enviado pela aba de contato.

Figura 79: Janela de especificações aberta ao se pressionar o botão

A aba de Contato permite ao usuário se comunicar diretamente conosco via e-mail. Primeiramente ele caracterizava o motivo do seu contato por "problema no software", "problema em um port", "elogio", "sugestão" ou "outros". Depois ele fornecia uma descrição mais detalhada do motivo do contato, e então seu endereço de e-mail.

Com isso o usuário poderia entrar em contato conosco de maneira simples e rápida, mas isso nos levou a dois problemas: o primeiro era o usuário não saber qual a categoria do seu problema, e o segundo era o usuário não se identificar (não

mencionar o próprio nome), o que dificultava a comunicação. Para resolver o problema, a seleção de motivo de contato foi substituída por um campo de assunto, e um campo de nome foi adicionado.

Voltando à tela principal, a aba de Doar é uma área reservada para doações. Ela possui 4 botões, os quais permitem que as pessoas auxiliem o projeto com doações que podem se destinar a:

- **WineHQ:** Comunidade Wine, responsável pelo motor usado pelo Wineskin.
- **doh123:** Criadora do Wineskin, responsável pela tecnologia de wrappers.
- **Paulthetall:** Financiador e divulgador do Porting Kit, e maior criador de WSIs.
- **Vitor:** Eu, o programador, gerenciador da base, webmaster e webdesigner.

6.2.6 - Criação de Wrapper com WSI

O mecanismo principal do Porting Kit. Utilizando os arquivos WSI previamente descritos, o Porting Kit é capaz de recriar um wrapper seguindo suas instruções. A criação de um wrapper por meio deste consiste em várias etapas realizadas em sequência, mas o principal fator nesse processo é justamente a simplicidade.

Durante a criação de um wrapper com o Wineskin puro, o usuário necessita de conhecimento para saber como criar o port para aquela aplicação, mas mesmo que ele esteja seguindo instruções prontas ele está sujeito a falhas, expostas pela interação do usuário com o programa. Em outras palavras, é necessário reduzir a interação com o usuário ao máximo, se possível resultando em interação praticamente zero. Esse é um dos ideais que o Porting Kit deseja alcançar.

6.2.6.1 - Instalação de client silenciosa

A primeira etapa realizada pelo Porting Kit é avaliar se um port necessita de um dos clients de DRM (Digital Rights Management) mais comuns: Steam, Origin ou Uplay. Isso ocorre pois aqueles que se enquadram nessa categoria possuem pré-configurações específicas. A necessidade de instalar um client pode ser

identificada pelo nome do WSI ou por uma variável dentro do WSI. A instalação destes softwares só é feita após a primeira fase de instalação dos winetricks.

Primeiramente se verifica a variável, a qual passou a ser usada na versão 1.7.93; uma booleana que se for verdadeira permite a instalação e se for falso nega, e é atribuída durante a criação do WSI. Caso a variável não exista, o nome do WSI será a referência.

Se ele terminar com ‘(Origin)’ ou começar com ‘Originbuild’, então é um port com Origin. Se terminar com ‘(Steam)’ ou começar com ‘Steambuild’, então é um port com Steam. O mesmo sistema se aplica a ports com Uplay, e a verificação desconsidera o *case* das letras.

Por muito tempo, a instalação de clients por parte do Porting Kit era apenas composta pelo download e instalação do client, obtendo-o de um link direto oficial para a versão mais recente. A partir da versão 1.8.52 a instalação do Steam passou a acompanhar o seu winetrick dependente (*nocrashdialog*), ou seja, o que ele requer para funcionar propriamente; o mesmo se aplica ao Origin na versão 1.8.61 com os winetricks *msxml3*, *vcrun2013*, *vb6run*, *nocrashdialog*, *vcrun2010* e *macdriver=mac*, e ao Uplay na versão 1.8.66 com os winetricks *crypt32*, *gdiplus_winxp*, *msxml3* e *winhttp*.

Isso resolvia muitos problemas, mas era preciso mais. Assim como na instalação de winetricks (Capítulo 6.2.6.5) era necessário realizar a instalação de forma silenciosa para evitar problemas ocasionados pela interação com o usuário. A melhor maneira de resolver isso seria chamar os instaladores com suas flags de instalação silenciosa.

O primeiro problema foi descobrir quais eram essas flags, pois esse não é o tipo de informação disponibilizada abertamente. Por sorte já havia conhecimento sobre o projeto da aplicação de Windows *Chocolatey*, o qual tem como objetivo ser uma alternativa para o *apt-get* (programa que baixa e instala aplicações variadas via linha de comando) do Linux no Windows. Para realizar tal feito, o *Chocolatey* possui em seus scripts os argumentos para a instalação silenciosa de diversos apps, inclusive os três mencionados.

O segundo problema foi descobrir como usar essas flags, pois o Wineskin não suporta o uso de flags na execução direta de programas (via linha de

comando). A única forma de usar flags em um executável do Windows diretamente seria definir o executável do instalador e suas flags como o executável principal do port temporariamente, mas isso poderia causas complicações posteriores caso houvesse uma interrupção na instalação. Sendo assim, a linha de comando era a única opção plausível.

O uso de um arquivo *.bat* foi a primeira alternativa, pois ele poderia ser chamado pelo Wineskin e chamar um executável com flags. O problema é que este arquivo abria uma janela do prompt de comando durante a sua instalação, o que abria a possibilidade para falhas, permitindo que o usuário fechasse a janela, cancelando a operação.

Em seguida tentou-se usar um arquivo *.vbs*, e depois de vários testes conseguiu-se criar um capaz de executar o instalador corretamente evitando interações do usuário; isto é, na maioria dos clients.

Apesar dos instaladores do Uplay (flags: */S /NCRC*) e do Steam (flag: */S*) terem se tornado completamente silenciosos, o instalador do Origin (flag: */S*) continuou a mostrar a janela para aceitar os termos de uso e o erro de atualização. Além disso, não foi possível encontrar os parâmetros que definiam os valores de suas caixas de seleção durante o processo de instalação (o que era necessário para desabilitar a atualização automática do Origin e seus ports, o que costuma danifica-los), por isso foi necessário alterar o resultado das mesmas manualmente.

Comparando dois wrappers equivalentes, cuja única diferença era a marcação ou não das caixas, foi possível descobrir que isso era nada mais nada menos do que campos no registro do Windows. Dentro do registro de sistema, na seção *[Software\\Origin]*, as variáveis *AutopatchGlobal*, *Autoupdate* e *TelemOO* inverteram seus valores booleanos. Consequentemente, essa correção passou a ser feita manualmente após a instalação do Origin. Ainda não foi possível resolver o problema das janelas abertas pelo instalador do mesmo.

Posteriormente descobriu-se que os arquivos *.vbs* não era executados corretamente em todos os motores do Wineskin, então voltou-se a usar os arquivos *.bat.* A janela de prompt era menos vulnerável a ações do usuário do que as janelas comuns de instalação, mas ainda era passível de falha humana.

Para resolver a situação, o código-fonte do Wineskin foi analisado, pois o mesmo era capaz de executar arquivos *.bat* sem a janela de prompt fosse mostrada com a marcação de uma caixa de seleção. Foi descoberto então que se uma instrução fosse executada precedida do comando *start*, o qual era viável devido ao executável start.exe contido em todo wrapper, ela seria executada paralelamente ao prompt em vez de diretamente nele.

Inicialmente pensou-se em chamar o arquivo *.bat* com esse executável, mas então volta-se ao problema das flags, o que requisitaria um outro arquivo *.bat* com essa chamada. Pensando-se nisso, o caminho do instalador e sua flags passaram a ser chamados diretamente com o comando *start*, que era então colocado em um arquivo *.bat*. A conclusão é que a janela de prompt passou a surgir por apenas um instante, e então desaparecer, o que era bem melhor do que mantê-la durante toda a instalação, e evitava erros por parte do usuário. Os problemas na instalação do Origin se mantiveram mesmo após tudo isso.

Desde a versão 2.6 do Porting Kit, os scripts para instalação de clients de DRMs passaram a ser lidos de um arquivo JSON hospedado no servidor de portingkit.com, permitindo que façamos modificações no script sempre que necessário, sem precisar lançar uma nova atualização para o Porting Kit sempre que uma mudança assim surgir.

6.2.6.2 - Verificação de legitimidade

Em seguida é verificada a legitimidade e compatibilidade do WSI. Primeiro avalia-se se a versão do WSI em questão é compatível (WSIs muito antigos não são suportados devido a suas múltiplas diferenças). Caso seja compatível, é avaliado se o dito WSI foi baixado de nosso servidor ou se ele é local. Se ele tiver sido baixado do servidor mas não tiver uma assinatura válida, o Porting Kit irá contra recomenda-lo. Se não tiver sido baixado do servidor, o usuário receberá um alerta de que o Porting Kit não se responsabilizará por que qualquer dano que ele possa causar.

6.2.6.3 - Atualização de Componentes

Uma vez que a instalação seja confirmada, o Porting Kit irá conferir se o Wrapper Mestre do Wineskin está atualizado. O Wrapper Mestre é o wrapper usado como modelo para a criação de qualquer wrapper Wineskin; mantê-lo atualizado é sempre necessário para garantir que a aplicação irá funcionar em sistemas mais recentes. Utilizando o nome do Wrapper Mestre que se encontra em seu diretório padrão do Wineskin, é possível obter sua versão e compará-la com a mais recente disponível no site do Wineskin.

A etapa seguinte consiste em verificar se o motor que será usado está disponível. No caso de motores originais do Wineskin verifica-se a pasta de motores, e caso ele não seja encontrado é baixado por um link direto. Caso seja um motor personalizado, verifica-se a pasta de motores para ver se ele já está lá, e caso contrário ele é baixado com o nome que lhe foi designado.

6.2.6.4 - Construindo o Wrapper

Depois que o Wrapper Mestre e o motor são verificados, o Wrapper Mestre é copiado para o diretório de destino do novo wrapper com o nome do mesmo, e o motor é copiado para dentro deste. Então, por meio do Wineskin interno do wrapper, executado por meio de linha de comando, é realizado um Wineprefixcreate, atualizando as configurações do wrapper e adaptando-o ao motor escolhido.

Durante essa etapa, o usuário é questionado se deseja instalar o Gecko ou o Mono em seu wrapper, mas estes não foram úteis em nenhum dos wrapper desenvolvidos. Houve uma tentativa de remover esses diálogos em outubro de 2015, mas a remoção não funcionava sempre, então o máxima que se pôde fazer foi deixar um alerta na área de progresso avisando ao usuário para não instalá-los.

Agora a etapa mais esperada: a configuração via WSI. Aqui, todas as outras informações contidas no WSI se tornam palpáveis, e são aplicadas no wrapper: configurações avançadas no Wineskin, ativação/desativação do Mac Driver (motor gráfico desenvolvido no Wine que não utiliza o WineskinX11), menu do wrapper, decoração de janelas, ativação/desativação do Direct3D Boost (útil para melhor a

performance de games feitos em DirectX), drives, extensões suportadas, instalações de winetricks (as quais eram como os instaladores de Wineskin, mas se tornaram silenciosos em outubro de 2015; um mês antes, esse recurso também passou a abranger a troca de motor em tempo de instalação), registros de Direct3D, OpenGL e sobreposições de DLLs e então a configuração dos executáveis, aparência do wrapper e instaladores, nos quais a ordem varia.

No passado, uma string de copyright também era salva no WSI para ser aplicada no wrapper, no entanto para evitar a disseminação de cópias ilegítimas de ports criados pelo Porting Kit, isso foi removido para dar lugar a uma string de copyright fixa que remete ao programa.

6.2.6.5 - Instalação de Winetricks Silenciosa

Inicialmente a instalação dos winetricks necessitava de interações do usuário para ser concluída (com os diálogos de instalação padrões do Windows), mas durante as tentativas de instalação do CytoClus (Capítulo 5.1) a professora Myrian Costa teve complicações durante os diálogos dos instaladores, o que levou a questão: *E se outros usuários comuns enfrentarem o mesmo tipo de dificuldade?*

Para corrigir isso, decidiu-se que as instalações passariam a ser silenciosas (*silent, unattended*; sem interação direta com o usuário) para evitar essas complicações. Primeiro, ao se descobrir que o arquivo de winetricks (*winetricks.sh*) já possuía as flags para o modo silencioso de todos os seus programas, cogitou-se mudar os argumentos durante a instalação dos winetricks para tentar usar esses argumentos nas instalações.

O Wine ativa as instalações silenciosas por meio das flags *-q* ou *--unattended*, mas se chegou à conclusão de que o Wineskin não poderia passar outros argumentos durante a instalação de winetricks que não fossem os próprios nomes dos winetricks, impedindo que uma dessas flags fosse passada.

Analisando-se o arquivo de winetricks, foi possível notar que a única ação das flags era atribuir o valor 1 para a variável *winetricks_set_unattended*, e o resto do arquivo usaria essa variável como um número *booleano* para decidir se instalaria de forma silenciosa ou não.

Foi observado então que o Wineskin passava uma flag durante a instalação de winetricks: *--no-isolate*, a qual definia o valor de WINETRICKS_OPT_SHAREDPREFIX como 1, que basicamente significa que os winetricks devem ser instalados no Wineprefix do wrapper, um pré-requisito para o funcionamento correto do Wineskin. Sendo assim, se a linha desta flag fosse alterada seria possível tornar toda instalação também silenciosa, e isso levou a uma atualização manual do arquivo efetuada pelo Porting Kit:

Linha original: *--no-isolate) WINETRICKS_OPT_SHAREDPREFIX=1 ;;*

Linha alterada: *--no-isolate) WINETRICKS_OPT_SHAREDPREFIX=1 ; winetricks_set_unattended 1 ;;*

E isso resolveu o problema, tornando toda instalação de winetrick silenciosa.

6.2.6.6 - Configurando o Caminho de Execução

Para evitar problemas com usuários que realizam a instalação dos jogos e os executam ao finalizá-la, quando os jogos não possuem um caminho relativo (ou seja, indireto; que necessita do uso do caracter ‘*’ para ser encontrado) suas configurações de executáveis e a aparência do wrapper são definidas antes da instalação começar, logo após os primeiros winetricks e antes dos clients. No entanto, nos casos que o caminho varia, por falta de opção, não há como realizar essas configurações antes da instalação. Isso faz com que o programa seja instalado antes das configurações serem aplicadas.

Se as aplicações necessitarem de clients, os mesmos serão instalados logo após os winetricks pré-instalação. Arquivos também podem precisar ser baixados e até instalados para usar o programa desejado. Dentre os arquivos extras que podem ser baixados podemos dividi-los em suas categorias: os que vem antes do instalador do jogo e os que vem depois. Depois do instalador e dos arquivos extras são instalados os winetricks pós-instalação, necessários em alguns raros casos como antes era na instalação da aplicação da Origin.

6.2.6.7 - Reparo para Placas de Vídeo

Durante os testes com um computador que possuía uma placa de vídeo Intel Iris foi observado que a quantidade de memória virtual disponível não era obtida corretamente pelo Wineskin (originalmente acreditou-se que fosse responsabilidade do Wine) para placas deste tipo, o que fazia com que muitas aplicações ficassem restritas a uma memória de vídeo muito inferior à verdadeira, impedindo que certas aplicações (especialmente jogos modernos, como *The Elder Scrolls V: Skyrim*) fossem executadas.

Para resolver esse problema, em computadores com Intel Iris, a obtenção automática de memória de vídeo passou a ser desativada atribuindo o valor *false* para a chave *"Try To Use GPU Info"* do wrapper, e a memória passou a ser configurada manualmente no registro do wrapper, tendo seu valor obtido da chave *"VRAM (Dynamic, Max)"* do seguinte comando de terminal:

system_profiler SPDisplaysDataType

Com este valor em mãos, bastava passa-lo para Megabytes e colocá-lo na chave *VideoMemorySize* do registro de usuário *[Software\\Wine\\Direct3D]*. Isso resolveu o problema para os usuários de computadores Intel Iris, e posteriormente também dos usuários de Intel HD, nos quais o mesmo problema foi observado em alguns jogos.

Conforme o tempo passou, observou-se que haviam casos em placas NVIDIA e AMD também, levando à conclusão que se tratava de um problema muito mais comum e sério. Para conseguir criar um algoritmo que pudesse resolver o problema para todos os tipos de placa, foi requisitado no fórum que os usuários ajudassem, fornecendo a saída do comando citado anteriormente em seus computadores para análise.

Em menos de uma hora foram supridos dados suficientes para a criação do algoritmo. A lista de dispositivos de vídeo disponíveis é transformada em uma lista de dicionários, os quais são então ordenados pela chave “*Bus”* na seguinte ordem: *Built-In*, *PCI* e *PCIe*; desta forma os dispositivos são ordenados do menos potente

ao mais potente disponível. O último da lista é então retornado, e se obtém seu tamanho de memória das variáveis “*VRAM (Total)”* ou “*VRAM (Dynamic, Max)”*, dependendo de qual estiver disponível. Com isso, o método passou a ser usado durante a criação de ports em todos os tipos de computador.

6.2.6.8 - Configurações Finais

Neste momento era conferido se o wrapper possui um client, e se está usando o WineskinX11 em vez de Mac Driver. Isso era feito por que os wrappers com clients possuem o Wine System Tray, uma pequena janela que pode incomodar e às vezes trazer pequenos problemas. Para evitar isso, era oferecida ao usuário a possibilidade de remover essa janela se ele desejasse. Esse diálogo foi removido por que a maioria dos usuários não ligava para a presença ou não da janela, e se tornou uma opção na janela de Propriedades.

Por fim, é removida a lista de aplicativos que inicializam com o port, os quais apenas atrapalham durante uma mudança de motor ou durante um *Refresh* de wrapper. Agora, a réplica do wrapper está completa.

7. CONCLUSÃO

Partindo do princípio, Appleverse Wizard ainda não parecia possuir muito potencial, mas conseguiu chamar a atenção de alguns curiosos. Do dia 24/01/12, o momento em que foi hospedado no site Sourceforge, até o dia 12/01/15 o programa teve um total de 367 downloads. Um número pequeno se considerada a margem de 3 anos.

Devido a um problema com a hospedagem no Sourceforge, os números de download do Wigidrasil se perderam, mas não haveria muito a se esperar. Portanto, temos em seguida o Gamma Studio. Este conseguiu um total de 417 downloads de 28/02/13 até 12/01/15, o que é melhor que o Appleverse Wizard, mais ainda longe do aceitável.

E então chegamos ao Projeto Porting dos dias atuais, com o número de downloads entre 29/12/14, quando o mesmo foi anunciado oficialmente, e 12/01/15, data em que os dados citados foram obtidos. Durante esses 15 dias, o Porting Center atingiu a marcar de 13 downloads, o que não é muito chocante já que ele se tornou um programa voltado a desenvolvedores. No entanto, Porting Kit obteve neste mesmo tempo esmagadores 568 downloads e 1.135 execuções, ultrapassando as marcas de seus antecessores em meras duas semanas.

O crescimento no número de interessados no projeto mostra que ele evoluiu de uma aplicação de uso ocasional para um aplicativo rotineiro, onde a pessoa pode abrir e pensar *“Vamos ver o que temos de novo hoje”*, por que a cada dias programas e mais programas são portados e transferidos para o nosso servidor.

O sistema WSI-Servidor se tornou similar ao Software Livre praticado pela comunidade Linux em alguns aspectos: as pessoas podem auxiliar umas às outras livremente, usuários de mais influência conseguem disseminar informação mais rapidamente, e o uso do software não se restringe a ninguém.

Desta forma, o Projeto Porting pode ser usado livremente no meio acadêmico, e mesmo no meio empresarial, para facilitar a instalação de programas de Windows no macOS, expandindo a quantidade de usuários de um determinado software e dispensando a necessidade de se utilizar múltiplos sistemas por

necessidades mínimas. O Porting Kit pode provar seu valor pelo número crescente de usuários: o número de visitantes únicos em 2017 foi de 522.035, bem superior aos 316.796 visitantes de 2016, e aos 73.979 visitantes de 2015.

Apesar de conhecimentos sobre Wine ainda serem necessários para se criar um port, a geração de arquivos WSI2 dispensa conhecimento prévio, portanto uma vez que um port para uma aplicação tenha sido feito, o Projeto Porting pode facilitar a sua instalação em outros computadores Apple, independente de quais sejam, além de auxiliar a preservar as informações de como o port foi criado dentro de si.

7.1 - UPLOAD DE PORTS COM PAUL THE TALL

A parceria com Paul The Tall se mostrou proveitosa para ambos os lados. Antes da parceria, Paul possuía quase 1.000 ports em seu website. Hoje existem mais de 1.100 ports no Porting Kit, dos quais a grande maioria foi provida pelo próprio Paul, enquanto seu site hoje não possui mais nenhum port diretamente disponível para download, sem o uso do Porting Kit.

O fato de que o programa foi adotado ao ponto de se tornar a única forma de distribuição dos ports por Paul indica o quanto ele se mostrou mais versátil e mais confiável do que o antigo método de fazer upload dos ports inteiros para serviços de hospedagens pagos.

7.2 - RECEPÇÃO DO PÚBLICO

Depois de bastante tempo disponível para o público, o Porting Kit construiu uma boa reputação em meio a algumas comunidades na internet, como o Reddit (especialmente no subreddit */r/macgaming*) e o fórum da GOG.com.

O gráfico na figura 80 exibe a quantidade de instalações bem sucedidas realizadas com o Porting Kit de Setembro de 2017 a Março de 2019, o qual totaliza 229.884 instalações, de acordo com o Google Analytics.

Figura 80: Quantidade de instalações bem sucedidas via Porting Kit de acordo com o Google Analytics

Desde a versão do 2.7.40 do software, lançada em 24 de Agosto de 2017, o Google Analytics passou a ser usado para ficar de olho em potenciais bugs no software, além de permitir contabilizar quantos aplicativos eram instalados por meio do mesmo.

Além disso, desde a versão 2.2.76 (07/07/2016), o Porting Kit passou a requisitar aos usuários feedback sobre o programa depois de 5 dias (não necessariamente contínuos) de uso. Na versão 2.9.0 (07/10/2017), isso mudou para 5 dias, 30 dias e 100 dias. Devido a isso, vários feedbacks puderão ser coletados, incluindo os listados a seguir:

"This is an excellent app. I love being able to have all my old Windows games on my Mac, and having them run smoothly is a wonderful bonus. Please keep up the good work!" - Anônimo - 08/06/18

"Porting Kit is a very welcome service after having a hassle with Wine and Winebottler, thanks! [...]" - Hans August - 20/06/18

"I've been loving it! Very easy to use, with no bugs encountered so far!" - Mac Hyde - 23/09/18

"It's great I love it!" - Peter Wiens - 22/07/18

"Porting Kit ist the greatest discovery I made this year. Thanks for providing and further improving it!" - Anônimo - 16/12/2018

Ao todo, aproximadamente 850 feedbacks foram submetidos pelos usuários. Graças a eles: bugs foram corrigidos, e problemas em ports foram identificados de forma mais ágil.

8. REFERÊNCIAS

DEVILHUNTER. CXEx Project Shutdown + Source Code - CXEx (Member Project). **The Porting Team**, 8 fev. 2011. Disponível em: <http://portingteam.com/topic/4748-cxex-project-shutdown-source-code/>. Acesso em: 15 set. 2017.

GOOGLE. reCaptcha: Easy on Humans, Hard on Bots. Disponível em: <https://www.google.com/recaptcha/intro/invisible.html>. Acesso em: 14 jun. 2017.

JARRETTJAWN. 5 Cool Features of GOG Galaxy - Blogs - Gamepedia, **Gamepedia**, 8 Mai. 2015. Disponível em: <http://www.gamepedia.com/blogs/726-5-cool-features-of-gog-galaxy>. Acesso em: 15 mai. 2017.

ELECTRONIC ARTS INC. Origin, **Origin**. Disponível em: <https://www.origin.com/bra/pt-br/store/about>. Acesso em: 15 mai. 2017.

WINE. WineHQ - Sobre o Wine. Disponível em: <https://www.winehq.org/about>. Acesso em: 15 mai. 2017.

ORACLE. Oracle VM VirtualBox, **VirtualBox**. Disponível em: <https://www.virtualbox.org>. Acesso em: 15 mai. 2017.

CODEWEAVERS. Run Microsoft Windows software on Mac or Linux | CodeWeavers. Disponível em: <https://www.codeweavers.com/products>. Acesso em: 15 mai. 2017.

THE GIMP TEAM. GIMP - GNU Image Manipulation Program. Disponível em: <https://www.gimp.org>. Acesso em: 15 mai. 2017.

INKSCAPE. About | Inkscape. Disponível em: <https://inkscape.org/en/about/>. Acesso em: 15 mai. 2017.

ITERATE GMBH. Cyberduck | Libre FTP, SFTP, WebDAV, S3, Backblaze B2 & OpenStack Swift browser for Mac and Windows, **Cyberduck**. Disponível em: <https://cyberduck.io>. Acesso em: 15 mai. 2017.

TIM KOSSE. FileZilla - The free FTP solution, **FileZilla**. Disponível em: <https://filezilla-project.org>. Acesso em: 15 mai. 2017.

MIKESMASSIVEMESS. WineBottler | Run Windows-based Programs on a Mac, **WineBottler**. Disponível em: <http://winebottler.kronenberg.org>. Acesso em: 15 mai. 2017.

DOH123. Wineskin, **Wineskin**. Disponível em: <http://wineskin.urgesoftware.com/tiki-index.php>. Acesso em: 15 mai. 2017.

APPLE INC. qlmanage (1) Mac OS X Manual Page. Disponível em: <https://developer.apple.com/legacy/library/documentation/Darwin/Reference/ManPages/man1/qlmanage.1.html>. Acesso em: 15 mai. 2017.

SHINY FROG. Image2Icon - Your Mac. Your Icons., **Image2Icon**. Disponível em: <http://www.img2icnsapp.com>. Acesso em: 15 mai. 2017.

APPLE INC. Object Encoding. Disponível em: <https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/ObjectEncoding.html>. Acesso em: 15 mai. 2017.

CEDAR LAKE VENTURES, INC. Vector Magic: Convert JPG, PNG images to SVG, EPS, AI vectors. Disponível em: <https://vectormagic.com>. Acesso em: 15 mai. 2017.

THE APACHE SOFTWARE FOUNDATION. SVG Rasterizer. Disponível em: <https://xmlgraphics.apache.org/batik/tools/rasterizer.html>. Acesso em: 15 mai. 2017.

MISCHA LUSTECK. SmillaEnlarger download. **SourceForge**. Disponível em: <https://sourceforge.net/projects/imageenlarger/>. Acesso em: 15 mai. 2017.

VALVEL CORPORATION. Steam, a plataforma definitiva de jogos online. Disponível em: <http://store.steampowered.com/about/>. Acesso em: 15 mai. 2017.

NEWZOO. Global games market report infographics 2013. Disponível em: <http://www.newzoo.com/infographics/global-games-market-report-infographics/>. Acesso em: 17 ago. 2015.

W3SCHOOLS. OS platform statistics. Disponível em: <http://www.w3schools.com/browsers/browsers_os.asp>. Acesso em: 17 ago. 2015.

INTERNET LIVE STATS. Number of internet users. Disponível em: <http://www.internetlivestats.com/internet-users/>. Acesso em: 17 ago. 2015.

COHEN, Peter. 'Cider' makes Windows games run on Intel Macs. **MacWorld**, 3 ago. 2006. Disponível em: <http://www.macworld.com/article/1052176/cider.html>. Acesso em: 9 dez. 2015.

CODEWEAVERS. CrossOver Mac and Linux changelog. Disponível em: <https://www.codeweavers.com/products/more-information/changelog>. Acesso em: 9 dez. 2015.

MACUPDATE. CrossOver Chromium for Mac, 15 set. 2008. Disponível em: <http://www.macupdate.com/app/mac/28784/crossover-chromium>. Acesso em: 9 dez. 2015.

CODEWEAVERS. CrossOver Chromium. Disponível em: <https://web.archive.org/web/20080916074136/http://www.codeweavers.com/services/ports/chromium/>. Acesso em: 9 dez. 2015.

THEDOCTOR45. Final issue with CrossOver wrapper, 13 abr. 2009. Disponível em: <https://web.archive.org/web/20090627185427/http://dev.ibrain.com.ua/tag/wrapper/>. Acesso em: 9 dez. 2015.

THEDOCTOR45. CrossOver Game wrapper tutorial. **MacPortingTeam**, 21 jun. 2009. Disponível em: <https://www.youtube.com/watch?v=gsmoJhTBFxM>. Acesso em: 9 dez. 2015.

DEVILHUNTER. CrossOver (CX) ports and Snow Leopard. **The Porting Team**, 16 ago. 2009. Disponível em: <https://web.archive.org/web/20100905061451/http://portingteam.com/2009/08/crossover-cx-ports-and-snow-leopard/>. Acesso em: 9 dez. 2015.

DOH123. OSXDosboxWrapper 1.0. **The Porting Team [Forum]**, 20 jun. 2009. Disponível em: <http://portingteam.com/topic/559-osxdosboxwrapper-10/>. Acesso em: 9 dez. 2015.

DOH123. WineWrapper 0.9.2. **The Porting Team**, 5 jul. 2009. Disponível em: <https://web.archive.org/web/20111106192914/http://portingteam.com/2009/07/wineskin-0-9-2/>. Acesso em: 9 dez. 2015.

DOH123. CXSkin Games 8, beta available!. **The Porting Team**, 27 set. 2009. Disponível em: <https://web.archive.org/web/20111009062537/http://portingteam.com/2009/09/cxskin-games-8/>. Acesso em: 9 dez. 2015.

THEDOH123. Wineskin. **InsanelyMac [Forum]**, 23 set. 2009. Disponível em: <http://www.insanelymac.com/forum/topic/188350-wineskin/>. Acesso em: 9 dez. 2015.

BULAREO. OS X - How to play wine wrapped game?? (CSX). **MacRumors [Forum]**, 11 nov. 2012. Disponível em: <http://forums.macrumors.com/threads/how-to-play-wine-wrapped-game-csx.1487640/>. Acesso em: 9 dez. 2015.

DOH123. CXZ. **The Porting Team**, 13 nov. 2009. Disponível em: <https://web.archive.org/web/20111009062859/http://portingteam.com/2009/11/cxz/>. Acesso em: 9 dez. 2015.

DOH123. CXS (10.4, 10.5, 10.6 compatible wrappers). **The Game Porting Team [Forum]**, 26 set. 2009. Disponível em: <https://web.archive.org/web/20091114145637/http://forum.portingteam.com/viewtopic.php?f=56&t=1669>. Acesso em: 9 dez. 2015.

A GEEK'S CHANNEL. Convert CX game to CXZ, 9 set. 2010. Disponível em: <https://www.youtube.com/watch?v=rC09mq0xbDk>. Acesso em: 9 dez. 2015.

BALITSKY, Aaron. How to port games to mac with CXZ (only Snow Leopard, not supported on Lion), 23 jan. 2011. Disponível em: <https://www.youtube.com/watch?v=iYt_nbQD_QU>. Acesso em: 9 dez. 2015.

DEVILHUNTER. CXZ wrappers & engines. **The Porting Team**, 21 nov. 2009. Disponível em: <https://web.archive.org/web/20100218180229/http://portingteam.com/2009/11/cxz-wrappers-engines>. Acesso em: 9 dez. 2015.

DEVILHUNTER. CXEx Central. **The Porting Team [Forum]**, 2 dez. 2010. Disponível em: <http://portingteam.com/topic/4259-sticky-cxex-central/>. Acesso em: 9 dez. 2015.

DRAKULIX. CiderX - Release Topic. **The Porting Team [Forum]**, 9 set. 2010. Disponível em: <http://portingteam.com/topic/3591-ciderx-release-topic/>. Acesso em: 9 dez. 2015.

CHRISTOFF. Bordeaux for Mac OS X. **The Bordeaux Technology Group**, 3 set. 2010. Disponível em: <http://bordeauxgroup.com/store/bordeaux-for-macosx>. Acesso em: 9 dez. 2015.

NJCLARK. Bordeaux for Mac OS X. **The Bordeaux Technology Group**, 6 mar. 2010. Disponível em: <http://web.archive.org/web/20100422135750/http://bordeauxgroup.com/store/bordeaux-for-macosx>. Acesso em: 9 dez. 2015.

DOH123. What is "Wineskin Pro" by The Bordeaux Group. **Wineskin**. Disponível em: <http://wineskin.urgesoftware.com/tiki-index.php?page=What+is+Wineskin+Pro+by+The+Bordeaux+Group>. Acesso em: 9 dez. 2015.

DAN, Craciun. Interview with Tom Wickline of Bordeaux Technology Group. **TuxArena**, 21 jun. 2011. Disponível em: <http://www.tuxarena.com/2011/06/interview-with-tom-wickline-of-bordeaux-technology-group/>. Acesso em: 9 dez. 2015.

CHRISTOFF. Wineskin Pro. **The Bordeaux Technology Group**, 28 mar. 2011. Disponível em: <http://web.archive.org/web/20110509075303/http://www.bordeauxgroup.com/store/nordeaux-software/wineskin-pro>. Acesso em: 9 dez. 2015.

THE BORDEAUX TECHNOLOGY GROUP. Wineskin Pro readme, 17 mar. 2011. Disponível em: <http://web.archive.org/web/20110609074047/http://www.bordeauxgroup.com/mac/wineskin-readme>. Acesso em: 9 dez. 2015.

VENOM88. WinOnX alternatives and similar software. **AlternativeTo**, Nov. 2011. Disponível em: <http://alternativeto.net/software/winonx/>. Acesso em: 9 dez. 2015.

EL-EMAM, Hisham. WinOnX. **Mac App Store**, 7 set. 2015. Disponível em: <https://itunes.apple.com/us/app/winonx/id421346233>. Acesso em: 9 dez. 2015.

DOH123. [Poll] WS4 engines. **The Porting Team [Forum]**, 25 jul. 2010. Disponível em: <http://portingteam.com/topic/3214-poll-ws4-engines/>. Acesso em: 7 jun. 2017.

DOH123. Mac OS X 10.8.2, don’t do it!. **Wineskin**, 20 set. 2012. Disponível em: <http://wineskin.urgesoftware.com/tiki-view_blog_post.php?postId=61>. Acesso em: 9 dez. 2015.

DOH123. OS X 10.8.2 and OS X 10.7.5. **Wineskin**, 21 set. 2012. Disponível em: <http://wineskin.urgesoftware.com/tiki-view_blog_post.php?postId=62>. Acesso em: 9 dez. 2015.

DOH123. What computers can run Wineskin. **Wineskin**. Disponível em: <http://wineskin.urgesoftware.com/tiki-index.php?page=What+Computers+can+run+Wineskin>. Acesso em: 7 jun. 2017.

DOH123. Wineskin 2.0 Release!. **The Porting Team [Forum]**, 8 fev. 2011. Disponível em: <http://portingteam.com/topic/4746-wineskin-20-release/>. Acesso em: 7 jun. 2017.

DOH123. Wineskin 2.5.5 Released!. **Wineskin**, 18 abr. 2012. Disponível em: <http://wineskin.urgesoftware.com/tiki-view_blog_post.php?postId=61>. Acesso em: 7 jun. 2017.

DOH123. Wineskin 2.5.8 Released!. **Wineskin**, 21 set. 2012. Disponível em: <http://wineskin.urgesoftware.com/tiki-view_blog_post.php?postId=63>. Acesso em: 7 jun. 2017.

DOH123. Wineskin 2.6.1. **Wineskin**, 4 out. 2015. Disponível em: <http://wineskin.urgesoftware.com/tiki-view_blog_post.php?postId=82>. Acesso em: 7 jun. 2017.

APPLE. Atualização do OS X Mountain Lion v10.8.2, 19 set. 2012. Disponível em: <https://support.apple.com/kb/DL1580>. Acesso em: 9 dez. 2015.

XQUARTZ. XQuartz. Disponível em: <http://www.xquartz.org>. Acesso em: 9 dez. 2015.

TRANSGAMING. NVIDIA to acquire Transgaming's cross-platform portability technology, 11 jun. 2015. Disponível em: <http://www.transgaming.com/nvidia-to-acquire-transgamings-cross-platform-portability-technology/>. Acesso em: 9 dez. 2015.

NUMBERUNO. CXZ engines installer v1.5 (обновлен 6.11.10). **TorrentMac**, 6 nov. 2010. Disponível em: <http://www.torrentmac.org/viewtopic.php?p=57579>. Acesso em: 21 dez. 2015.

DEVILHUNTER. CXZ Wrappers & Engines (download/how to here). **The Porting Team [Forum]**, 2 jun. 2011. Disponível em: <http://portingteam.com/topic/5457-sticky-cxz-wrappers-engines-downloadhow-to-here/>. Acesso em: 21 dez. 2015.

DOH123. Questions about Wineskin and about the Porting Community [E-mail]. Mensagem para: Vitor Marques de Miranda, 2 dez. 2015.

CODEWEAVERS. Licensing Questions. Disponível em: <https://www.codeweavers.com/store/licensing>. Acesso em: 15 mai. 2017.

CODEWEAVERS. CrossOver Pricing. Disponível em: <https://www.codeweavers.com/store/pricing>. Acesso em: 15 mai. 2017.

VITORMM. If your computer has an AMD or a NVIDIA video card, we need your help. **PORTING COMMUNITY [Forum]**, 6 jun. 2016. Disponível em <http://portingkit.com/smf/index.php?topic=189>. Acesso em: 29 mai. 2018.

9. GLOSSÁRIO

AX - Extensão usada por diversas aplicações de playback de video, mas quase sempre é associada ao arquivo de filtros do Microsoft DirectShow. Esse formato de arquivo permite a um usuário visualizar vídeos da Internet em tempo real sem ter que fazer o download do arquivo inteiro em seu disco local e assistir o vídeo apenas depois disso.

DESKTOP VIRTUAL - Método pelo qual um port é executado possuindo o seu próprio Desktop em forma de janela, seja em tela cheia ou não.

DIRECTX - Pacote de bibliotecas criadas pela Microsoft para renderização de conteúdos visuais.

DIRECTX9 - Winetrick responsável pela instalação do pacote de bibliotecas DirectX 9.0.

DLL - Dinamic-Link Library. Extensão comum de biblioteca do sistema operacional Windows.

DRIVE - Dispositivo de dados presente em um computador ou port. Exemplos: HD, leitor de CD/DVD, leitor de disquete, etc.

DRM - Digital Rights Management. Serviço que lhe permite ter acesso a conteúdos digitais que você tenha direito de acessar, como filmes e jogos.

D3DX9_36 - Winetrick responsável pela instalação da biblioteca d3dx9_36.dll, que faz parte do pacote de bibliotecas do DirectX 9.0.

ENGINE - Motor utilizado por ports para executar aplicações de Windows, transferindo os pedidos do programa para o sistema operacional nativo, fazendo-o ser usado como se estivesse no ambiente Windows.

ICNS - Extensão dos ícones usados no sistema operacional Mac OS X.

ICO - Extensão dos ícones usados no sistema operacional Windows.

JAVA - Linguagem computacional orientada a objeto, na qual o computador que utiliza um programa compilado na mesma precisa ter instalado o JVM (Java Virtual Machine).

MÁQUINA VIRTUAL - Emulador de um sistema operacional completo dentro de outro sistema, permitindo assim o uso de mais de um sistema operacional simultaneamente.

OBJECTIVE-C - Linguagem computacional voltada para o desenvolvimento de programas para Mac OS X e iOS.

OCX - OLE Control Extension (OCX). Extensão de biblioteca para Windows que contém controles do ActiveX.

PHYSX - Winetrick para instalação das bibliotecas proprietárias PhysX, as quais são muito usadas como motor para a física em jogos.

PIXEALIZADA - Adjetivo para uma imagem que possua resolução baixa o suficiente para que se possa distinguir o pixels nitidamente.

PORT - Pasta em forma de aplicação para Mac na qual é armazenada uma aplicação de Windows completa e o motor para a sua execução, ou um link para o mesmo.

RASTERIZAÇÃO - Processo pelo qual uma imagem vetorizada se torna uma imagem comum.

SVG - Extensão comum em imagens vetorizadas.

VETORIZAÇÃO - Processo pelo qual uma imagem comum se torna uma imagem vetorizada.

WINETRICK - Instrução passada para um port com a qual ele consegue simular as ações de bibliotecas específicas ou realizar certas alterações no sistema do port.

WRAPPER - “Embrulho" no qual uma aplicação de Windows pode ser instalada no Mac utilizando tecnologia baseada em Wine.

XML - Extensible Markup Language. Arquitetura de texto na qual o conteúdo de um documento se torna legível para humanos e máquinas seguindo diversas especificações.

10. ANEXOS

Anexo A: NOTAS DE LANÇAMENTO DO PORTING CENTER *(PORTING CENTER RELEASE NOTES)*
As alterações estão listadas da mais recente para a mais antiga.

http://portingkit.com/center/releasenotes.html

Anexo B: NOTAS DE LANÇAMENTO DO PORTING KIT *(PORTING KIT RELEASE NOTES)*
As alterações estão listadas da mais recente para a mais antiga.

http://portingkit.com/kit/releasenotes.html
http://portingkit.com/kit/releasenotes-1.html