CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Jogos Digitais

Vinícius Franco

AS VANTAGENS DO PYGAME NO DESENVOLVIMENTO DE JOGOS

Americana, SP

2014

CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Jogos Digitais

Vinícius Franco

AS VANTAGENS DO PYGAME NO DESENVOLVIMENTO DE JOGOS

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Jogos Digitais, sob a orientação do Prof.. Esp. Marcos Francisco Pereira da Silva

Área de concentração: Jogos digitais.

Americana, SP

2014

Franco, Vinícius

F894v As vantagens do *Pygame* no desenvolvimento de jogos. / Vinícius Franco. – Americana: 2014.

37f.

Monografia (Graduação em Tecnologia em Jogos digitais). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.

Orientador: Prof. Esp. Marcos Francisco Pereira da Silva

1. 2. Desenvolvimento de software I. Silva, Marcos Francisco Pereira II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.3.05

AS VANTAGENS DO PYGAME NO DESENVOLVIMENTO DE JOGOS

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Jogos Digitais pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.
Área de concentração: Jogos digitais.

Americana, 04 de Dezembro de 2014.

Banca Examinadora:

Marcos Francisco Pereira da Silva (Presidente)
Especialista
FATEC Americana

Francesco Artur Perrotti (Membro)
Mestre
FATEC Americana

José Luiz Rondelli (Membro)
Graduado
FATEC Americana

RESUMO

Este estudo tem a finalidade de mostrar aos desenvolvedores de jogos as vantagens da utilização da biblioteca Pygame destinada à criação de jogos no Python. Apresenta-se aqui, a linguagem de programação Python, abordando desde assuntos históricos até os mais técnicos, que fornecem uma introdução a sintaxe e demonstram as diferenças entre as versões atualmente disponíveis. Em um outro momento o trabalho é destinado ao estudo do Pygame, mostrando sua história, suas características e alguns jogos de qualidade profissional que utilizam esta ferramenta com sucesso. Para compreender um pouco mais a fundo a biblioteca Pygame, foi desenvolvido um jogo de plataforma 2D que como resultado demonstra alguns conceitos desta ferramenta em funcionamento prático, e como algumas funções auxiliam o desenvolvimento da lógica do *Game Loop*. Por fim, apresenta-se uma analise dos pontos vantajosos da utilização do Pygame e porque empresas que possuem um maior investimento podem optar por outras ferramentas de desenvolvimento.

Palavras Chave: jogos; python; pygame;

ABSTRACT

This study has the purpose to show to games developers the advantages of using Pygame library intended to Python games development. It is presented the Python programming language, showing from historical subjects to the technical matters, to provide an introduction to the syntax and present the differences between the versions currently available. In another moment, this work is intended to the Pygame study, presenting its history, features and some example games with professional quality that uses the library with success. To understand a little deeper about Pygame, a platform 2D game was developed, which result shows some concepts of this tool in practical work, and how some functions help the development of Game Loop concept. At the end, it shows an analysis of the advantages of Pygame usage and why companies with a higher investment may prefer another development tools.

Keywords: *games; python; pygame*

SUMÁRIO

LISTA DE FIGURAS E DE TABELAS

1 INTRODUÇÃO

Jogos digitais são uma parte importante do mercado de entretenimento nos dias de hoje, e essa importância atrai jogadores e profissionais de TI para atuar no ramo de desenvolvimento. Essa atração fez com que existissem diversas maneiras de desenvolver um jogo, *engines* e bibliotecas são criadas para que cada vez mais isso se torne simples.

Pygame é uma biblioteca da linguagem de programação Python cujo foco é o desenvolvimento de jogos de uma maneira fácil e otimizada, e neste trabalho foi utilizado como recurso para o desenvolvimento de um jogo em plataforma 2D.

O objetivo deste trabalho é exemplificar alguns recursos do Python e de sua biblioteca Pygame e como eles podem ser implementados em um jogo funcional, visando principalmente mostrar suas vantagens e desvantagens, apresentando uma nova alternativa para quem deseja se tornar um desenvolvedor de jogos. Também é explicado a lógica do *Game Loop,* assim como a lógica de um jogo em plataforma 2D.

O trabalho foi estruturado em três capítulos, sendo que o primeiro oferece uma introdução à linguagem de programação Python, apresentando sua história, sua sintaxe e explicando seus pontos fortes e fracos. O segundo conceitua a biblioteca Pygame, desde de sua história, a evolução ao longo dos anos e apresenta alguns grandes jogos que foram desenvolvidos utilizando esse recurso. O terceiro capítulo explica como o jogo foi desenvolvido utilizando os recursos, explicando as funções do Pygame e a estrutura de um jogo plataforma 2D. Com base nas informações adquiridas em estudos realizados nos capítulos anteriores, o capítulo seguinte se reserva às considerações finais.

2 O PYTHON

Segundo McKeown (2010), programação é a maneira que os humanos explicam para um computador determinadas regras com o intuito de realizar uma tarefa, normalmente utilizada para resolver problemas. Este método de comunicação entre programador e máquina é chamado de linguagem de programação, e segundo Tiobe (2014), o Python se destaca em oitavo lugar das linguagens mais utilizadas em 2014, também recebeu o prêmio *"Programming Language of the Year"* do ano de 2007 e 2010.

Todos os códigos fontes presentes neste trabalho seguem o conceito de que, o símbolo ">>>" significa *input* de uma linha ou bloco de código, "#" representa comentários no código e linhas sem prefixos são considerados *output* do Python.

Nesse capítulo foi apresentada uma introdução à linguagem de programação Python, e mais detalhes sobre a história e sua evolução com o passar do tempo.

2.1 O QUE É O PYTHON?

Em um breve resumo feito pela empresa responsável, "Python é uma linguagem de programação orientada a objeto, interativa e imperativa [...] Python combina um poder notável com uma sintaxe muito clara" (PYTHON SOFTWARE FOUNDATION, 2014a). O Python tem diversas semelhanças com a linguagem Perl, pela simplicidade de ambas as sintaxes. Pelo fato do Python ser *open source*[1], ele está disponível para ser melhorado utilizando C e C++.

Atualmente a linguagem tem seus diretos das versões 2.1 em diante reservados a Python Software Foundation, uma organização sem fins lucrativos, do qual o objetivo é evoluir a linguagem e propagar o uso da mesma (PYTHON SOFTWARE FOUNDATION, 2014a). Python é livre e pode ser usado para propósitos pessoais e comerciais livre de qualquer custo.

Uma das principais vantagens do Python e de suas bibliotecas serem multiplataforma, ou seja, é possível utilizar o mesmo código Python em vários sistemas operacionais diferentes, tais como Windows (MICROSOFT CORPORATION, 1985), Mac OS X (APPLE INC., 2001), AS/400 (IBM, 1988), BeOS

[1] "*Open source software* é um *software* que pode ser utilizado, modificado e compartilhado por qualquer pessoa, gratuitamente" (OPEN SOURCE INITIATIVE, 2014, Tradução nossa)

(BE INCORPORATED, 1996), MorphOS (THE MORPHOS DEVELOPMENT TEAM, 2000), MS-DOS (MICROSOFT CORPORATION, 1981), OS/2 (IBM; MICROSOFT CORPORATION, 1987), OS/390 (IBM, 1995), RISC OS (CASTLE TECHNOLOGY; RISC OS OPEN, 1987), Nokia Symbian (NOKIA, 1997), Solaris (ORACLE CORPORATION, 1992), Windows CE (MICROSOFT CORPORATION, 1996), HP-UX (HEWLETT-PACKARD, 1984), entre outros (PYTHON SOFTWARE FOUNDATION, 2014b).

Como afirmado por (TIOBE SOFTWARE, 2014), Python está em oitavo lugar na lista de linguagens de programação mais usadas, e isso faz com que algumas empresas tenham interesse na linguagem, bons exemplos disso são: Google, Yahoo Maps e NASA. No desenvolvimento de jogos destaca-se duas empresas, porém de ramos diferentes, a DICE, subsidiaria da Eletronic Arts, desenvolveu os jogos Battlefield 2 (DIGITAL ILLUSIONS CE, 2005) e Battlefield 2142 (DIGITAL ILLUSIONS CE, 2006) em Python. A outra empresa é a The Blender Foundation, responsável pelo *software* gratuito que possibilita a modelagem em 3D e animações, amplamente utilizada em jogos digitais (PYTHON SOFTWARE FOUNDATION, 2014c).

2.2 HISTORIA DO PYTHON

"Python foi criado no início da década de 1990 por Guido van Rossum na Stichting Mathematisch Centrum na Holanda como um sucessor de uma linguagem chamada ABC" (PYTHON SOFTWARE FOUNDATION, 2014a, Tradução nossa). A linguagem ABC, é interativa, desenvolvida originalmente com a intenção de levar a programação aos iniciantes. Uma de suas principais características são seus programas que tipicamente chegam a um quarto ou um quinto do espaço de armazenamento que o mesmo programa utiliza quando desenvolvido em linguagens como Pascal ou C. (PEMBERTON, 2012)

Figura 1 - Logo do Python

Fonte: Python Software Foundation (2007)

Como visto na Figura 1, o logo do Python mostra a imagem de duas cobras entrelaçadas, fazendo referência a família *Pythonidae* do reino dos animais. Embora o logo oficial do Python sejam répteis, seu fundador tinha uma ideia bem diferente quando criou o nome. Guido van Rossum é um grande fã de uma antiga série de comedia da BBC chamada Monty Python da década de 70, mas ele preferia um nome curto e único, então ele decidiu chamar a linguagem apenas de Python. (HETLAND, 2005)

2.3 EXEMPLOS DA SINTAXE DO PYTHON

Uma característica da sintaxe do Python é ser simples na codificação, e sua falta de símbolos comuns com outras linguagens de programação. Uma pessoa que aprende programar utilizando linguagens como C, Java ou Javascript, ao iniciar o aprendizado de Python primeiramente perceberá a falta do ";" no final de cada linha de código. Indo mais a fundo é possível notar a diferença de separação dos blocos de códigos, que no Python é feito apenas pela indentação ao invés das chaves.

Figura 2 - Exemplo da sintaxe e indentação

```
#Python 3.x
>>> def printHelloWorld(i):
        if i > 2:
            print(“Hello World”)

Hello World
```

Fonte: Autor (2014)

No exemplo acima foi declarada uma função que recebe um número inteiro e verifica se ele é maior que 2, caso ele seja então é mostrado "*Hello World*" na tela do console. Nota-se claramente a falta dos símbolos de ponto e vírgula e dos símbolos das chaves, também é mostrado como funciona a indentação do Python, tudo que estiver dentro de um novo bloco de código, deverá ser colocado a quatro espaços a frente da indentação anterior.

Variáveis também são declaradas de maneira diferente, em algumas outras linguagens como C e Java, por exemplo, é necessário declarar a variável e definir seu tipo, podendo ser por exemplo um *int*, *float*, *char*, *boolean*, etc (PINHEIRO, 2012) (SMITH, 2011). No Javascript variáveis são declaradas utilizando a palavra reservada *var*, podendo ser atribuídas qualquer valor a elas (W3SCHOOLS, 2014). No Python, não existe palavra reservada para declaração de variável, porém, a variável deve ter um valor atribuído logo na sua declaração para ser definido automaticamente.

Figura 3 - Exemplo de declaração de variáveis no Python

```
#Declara b como um boolean com o valor True
>>>b = True
#Declara i como um int com o valor 10
>>>i = 10
#Declara f como float com o valor 2.5
>>>f = 2.5
#Declara g como uma string com o valor "Python"
>>>g = "Python"
```

Fonte: Autor (2014)

A maioria das pessoas que nunca estudaram programação mas sabem falar inglês, conseguem entender que a variável *solarSystem* representada na figura 4, contém o nome de todos os planetas do nosso sistema solar, e que o comando *for* pode ser lido como “para cada planeta no sistema solar, imprima o nome dele”.

Figura 4 - Exemplo Sistema Solar Python

```
#Python 3.x
>>>solarSystem = [“Mercury”, “Venus”, “Earth”, “Mars”,\
“Jupiter”, “Saturn”, “Uranus”, “Neptune”]

>>>for planet in solarSystem:
       print(planet)

Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune
```

Fonte: Autor (2014)

2.4 PYTHON 2 E PYTHON 3

Alguns anos depois do lançamento da primeira versão do Python, houve a necessidade de aprimora-lo, foi então que em 16 de Outubro de 2000 foi lançado o Python 2.x, mais alguns anos de avanços e em 3 de dezembro de 2008 surgiu o Python 3.x. (PYTHON SOFTWARE FOUNDATION, 2014d)

Atualmente existem duas versões ativas do Python, sendo as mais recentes o Python 2.7.8, lançado em 02 de julho de 2014 e o Python 3.4.1, lançado em 19 de maio de 2014. (PYTHON SOFTWARE FOUNDATION, 2014e)

Em um breve resumo, pode ser dito que “Python 2.x é o legado, Python 3.x é o presente e o futuro da linguagem” (PYTHON SOFTWARE FOUNDATION, 2014e). O motivo de ainda existir as duas versões é pela compatibilidade de suas bibliotecas, já que algumas estão disponíveis apenas para o Python 2.x. Mas esta limitação está chegando ao fim, pois todos os novos recursos desenvolvidos para Python estão disponíveis exclusivamente para Python 3.x.

2.3.1 O que mudou no Python 3.x?

A primeira mudança notada pelos programadores é o *print* que de uma palavra reservada agora tornou-se uma função *print().*

Figura 5 - Exemplo Hello World em Python 2.x e Python 3.x

```
#Python 2.x
>>> print “Hello World”
Hello World

#Python 3.x
>>> print(“Hello World”)
Hello World
```

Fonte: Autor (2014)

As palavras *True, False* e *None*, agora são palavras reservadas, no Python 2.x você poderia atribuir valores a elas, assim como elas mesmas. Gerando erros nada comuns.

Figura 6 - Exemplo do bug de atribuição de valores às palavras chaves

```
#Python 2.x
>>> True = False #Atribuindo o valor False para True
>>> True == False #Verificando se True é igual à False
True #Retorna True

#Python 3.x
>>> True = False #Atribuindo o valor False para True
SyntaxError: can't assign to keyword
```

Fonte: Autor (2014)

A função *range()* agora retorna um objeto iterável, assumindo o lugar de sua antiga função aliada *xrange(),* apresentando economia de memória em relação a quando retornava uma lista na versão anterior.

Figura 7 - Exemplo da utilização da função range

```
#Python 2.x
>>>range(5)
[0, 1, 2, 3, 4]

#Python 3.x
>>>range(5)
range(0, 5)
```

Fonte: Autor (2014)

No Python 2.x, a divisão de dois números inteiros sempre resulta em um número inteiro arredondado para baixo. No Python 3.x isso foi alterado e inserido o operador `//` para representar a divisão inteira.

Figura 8 - Exemplo de divisão com Python 2.x e Python 3.x

```
#Python 2.x
>>>5 / 2
2

#Python 3.x
>>>5 / 2
2.5
>>>5 // 2
2
```

Fonte: Autor (2014)

As figuras anteriores apresentam alguns exemplos de alterações na sintaxe do Python que tiveram um grande impacto para os programadores, diversas outras alterações foram realizadas, modificando bibliotecas e otimizando recursos. (PYTHON SOFTWARE FOUNDATION, 2014f)

3 O PYGAME

A popularidade dos jogos digitais com o passar dos anos fez com que novos recursos fossem criados e os velhos se aprimorassem, e isso expandiu muito as possibilidades para os desenvolvedores de jogos. Um desses recursos é o Pygame, uma biblioteca da linguagem Python que segundo seu criador Shinners(s.d.) é fazer tarefas simples de maneira fácil e as complexas de maneira correta.

Nesse capítulo foi apresentado a história do Pygame, alguns fatos sobre o mesmo e alguns jogos profissionais desenvolvidos utilizando esta ferramenta.

3.1 O QUE É PYGAME?

Segundo o *site* oficial do desenvolvedor, Pygame é uma biblioteca do Python, multiplataforma, *open source* (PYGAME ORG, 2014a), que facilita o acesso a biblioteca SDL[2] que oferece acesso à *inputs* e *outputs* amplamente utilizados em computadores, tais como, teclado, *mouse*, placa gráfica e áudio (SIMPLE DIRECTMEDIA LAYER ORG, 2014), permitindo assim o desenvolvimento de jogos de maneira simplificada.

Pygame foi escrito por Pete Shinners e teve sua primeira versão lançada em 28 de outubro de 2000 (PYGAME ORG, 2009), doze dias após o lançamento do Python 2.x, e até hoje permanece ativo e sujeito a melhorias.

Figura 9 - Logo do Pygame

Fonte: Pygame Org(2014)

[2] Simple DirectMedia Layer

3.2 FATOS SOBRE O PYGAME

Em seu site principal, o Pygame (PYGAME ORG, 2014a) cita algumas de suas principais características, dando ênfase em sua portabilidade e simplicidade. Abaixo estão algumas delas.

- Não necessita do OpenGL[3].
- Pode ser facilmente usado com processadores de múltiplos núcleos.
- Utiliza C otimizado e código Assembly para funções principais.
- Fácil instalação, não necessita a instalação de diversos pacotes para funcionar em diversos sistemas operacionais.
- Portátil. Oficialmente suporta Windows (MICROSOFT CORPORATION, 1985), Windows CE (MICROSOFT CORPORATION, 1996), BeOS (BE INCORPORATED, 1996), Mac OS X (APPLE INC., 2001), Solaris (ORACLE CORPORATION, 1992), entre outros.
- É simples e fácil de usar.
- Entre outros.

3.3 JOGOS PROFISSIONAIS DESENVOLVIDOS COM PYGAME

Quando falamos de desenvolvimento de jogos utilizando alguma ferramenta especifica, logo nos questionamos sobre sua capacidade e o quão longe podemos chegar utilizando-a. Segundo Sweigart (2013), o Python é notado na área do desenvolvimento de *games*, abaixo a lista com alguns jogos de qualidade profissional desenvolvidos em Pygame.

3.3.1 Metin2

O Metin2 (WEBZEN YMIR, 2004) é ambientado na cultura oriental onde batalhas *online* são o seu forte, o jogador é capaz de lutar contra hordas de monstros inteligentes, lutar com outros jogadores, ou montar uma equipe para uma batalha entre reinos.

[3] “A principal ambiente de desenvolvimento de aplicativos gráficos portáteis e interativos em 2D e 3D” (OPENGL ORG, s.d, Tradução nossa)

Segundo Webzen Ymir Games, o jogo foi desenvolvido pela própria empresa de origem coreana em 2004, tendo o seu lançamento no ano seguinte. Metin2 por ser um jogo multijogadores em escala massiva, necessita de conexão estável com a Internet. Portanto, o jogo está disponível desde o dia de seu lançamento até a presente data. Atualmente, o jogo é distribuído em 22 países incluindo o Brasil (WEBZEN YMIR GAMES, 2012). Na figura abaixo, temos um exemplo de uma cena do jogo.

Figura 10 - Screenshot do jogo Metin 2

Fonte: Metin2 (2012)

3.3.2 Frets On Fire

Frets On Fire (KYÖSTILÄ, 2006) simula uma guitarra com os botões do teclado, conforme a música avança, notas musicais aparecem ao jogador como botões coloridos. O objetivo do jogador é realizar a sequencia de cores correta para que a música seja tocada corretamente.

O jogo é *open source,* ou seja, seu código fonte está disponível gratuitamente para quem se interessar em saber como o jogo foi criado, ou simplesmente melhorá-lo.

Foi desenvolvido em 2006 por quatro pessoas, sendo que apenas uma delas era responsável pela programação, o objetivo da criação desse jogo era participar da competição de desenvolvimento de jogos da Unreal Voodoo, no qual ganhou o primeiro lugar. (FRETS ON FIRE, s.d.). A Figura 11 apresenta um exemplo da jogabilidade do Frets On Fire.

Figura 11 - Screenshot do jogo Frets On Fire

Fonte: Frets On Fire (s.d.)

3.3.3 Unity of Command

Unity of Command (2X2 GAMES, 2012) é um jogo de guerra do gênero de estratégia baseada em turnos, ambientado nos anos 1942 e 1943. O objetivo do jogo é liderar o seu exercito, conquistando cidades enquanto avança pelo território.

Foi desenvolvido e publicado pela 2x2 Games, uma empresa croata que possuí apenas um único funcionário e lançado em 2012 para Windows e Mac (DESURA, 2012).

Atualmente o jogo está disponível para Linux e na plataforma Steam[4], também já foram lançados dois pacotes de expansão, o Red Turn e o Black Turn, ambos fornecem recursos extras para o jogo. Abaixo na Figura 12 é apresentado uma das telas principais do Unity of Command.

Figura 12 - *Screenshot* do jogo Unity of Command

Fonte: 2x2 Games (2012)

[4] Steam é um serviço online disponibilizado pela Valve que permite aos assinantes o *download* ou acesso aos conteúdos de jogos. (STEAM, 2014)

Existem alguns outros jogos mencionados por Sweigart (2013) dos quais não foram mencionados nesse trabalho. A lista completa pode ser acessada em seu *site*[5].

[5] http://inventwithpython.com/blog/2013/02/19/what-professional-games-use-pygame/

4 O JOGO DESENVOLVIDO

Neste capítulo é apresentado com detalhes o jogo desenvolvido para obter experiência na biblioteca Pygame.

4.1 APRENDIZADO

Para aprender mais sobre a biblioteca Pygame foram utilizados de vários recursos, como a documentação do Pygame (2014), o livro "*Making Games with Python and Pygame*" de Sweigart (2012) e o site "*Program Arcade Games*" por Craven (2014), todos estão disponíveis gratuitamente *online*.

4.2 VISÃO GERAL

Mr.Carrot é um jogo de plataforma 2D desenvolvido pelo autor neste trabalho, no qual o objetivo é fazer o personagem principal lutar contra os monstros e desviar dos obstáculos, para salvar a princesa no final do cenário. O jogo utiliza a linguagem Python 3.4.1 e a biblioteca Pygame 1.9.2. Na figura abaixo é apresentado o início do jogo desenvolvido pelo autor.

Figura 13 - Mr.Carrot

Fonte: Autor (2014)

4.3 CONCEITOS BÁSICOS DO PYGAME

O vocabulário do Pygame para iniciantes em desenvolvimento de jogos pode parecer complicado, a seguir é apresentado alguns conceitos utilizados durante o desenvolvimento do caso de uso.

4.3.1 *Surface*

Surface que em português significa superfície, tem um tamanho pré-definido e é utilizada para desenhar qualquer imagem na tela, pode ser comparada à um quadro de pintura em branco. Ao iniciar essa classe, será criado um novo objeto de imagem que então será preenchida por um padrão em preto. (PYGAME ORG, 2014c)

4.3.2 *Sprite*

Um *sprite* representa qualquer objeto visível no jogo, é facilmente adicionado e removido do projeto. Também possui uma diversidade de funções que auxiliam a detecção de colisões. (PYGAME ORG, 2014d)

4.3.3 *Sprite Group*

Um *Sprite Group* é um objeto que pode conter uma parte ou todas as *sprites* do jogo, utilizado como uma lista de *sprites*. Esse objeto possui funções próprias como *pygame.sprite.Group.update()*, que chama a função *update()* de todos os *sprites* nele contido e *pygame.sprite.Group.draw(surface)*, que desenha todos os seus membros na *surface*. (PYGAME ORG, 2014d)

4.3.4 *Rect*

Rect no Pygame é tratado como objetos retangulares que podem ser utilizados para o posicionamento, dimensões do *Sprite* ou para detectar colisões. Pode ser criado oferecendo como argumento as coordenadas x, y e os valores de largura e altura do retângulo ou pode ser manipulado por objetos que possuam o atributo *rect*. (PYGAME ORG, 2014e)

Figura 14 - Exemplo de *Rect*

Fonte: Autor (2014)

O exemplo da figura 14, demonstra como um *Rect* está relacionado com a imagem, o princípio da colisão é que tudo o que entrar nesse retângulo estará colidindo com o jogador.

4.4 GAME LOOP

Jogos digitais assim como qualquer programa de interface gráfica não podem depender da entrada de informações do usuário (*inputs*) para continuar funcionando, se o jogador não realizar nenhum comando, o jogo ainda deve ser constantemente atualizado.

O conceito chamado de Game Loop, que independente da interação do jogador, continuará fazendo com que o jogo seja atualizado, de forma que o usuário recebe um retorno (*feedback*). Isso também permite controlar a velocidade do jogo, pois computadores mais antigo não possuem a mesma capacidade de processamento do que computadores atuais. (Nystrom, 2014)

O Game Loop é composto de três funções principais, conhecidas como *input*, *update* e *render.*

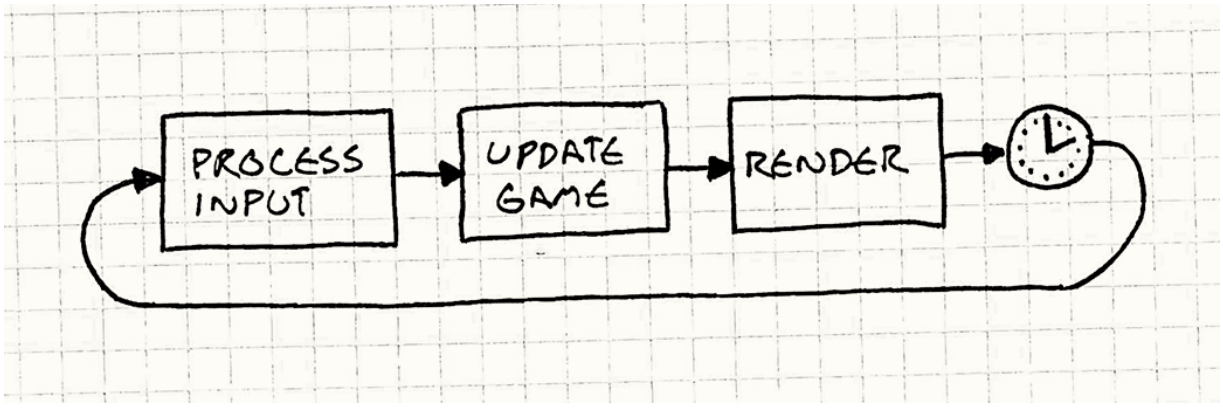
O método *input* é responsável por gerenciar todos os eventos que o usuário envia ao jogo, isso pode ser um movimento do *mouse*, o pressionamento de alguma tecla ou até mesmo quando o jogador fecha a janela do aplicativo.

O método *update*, atualiza todos os elementos do jogo que a ele são impostos, um exemplo disto são os pontos de vida e os pontos do jogador, que são atualizados sempre que o personagem entrar em contato com o inimigo ou encontrar algum item.

O método *render* desenha todos os *sprites* na tela de acordo com a ordem que é definido no projeto.

Para controlar o tempo que o jogo é atualizado, a unidade fps (*Frames per second* em português, quadros por segundo) é utilizada. A unidade divide o seu valor por um segundo e o resultado será o intervalo entre cada atualização do jogo, ou seja, quanto maior o seu valor, mais rapidamente o jogo será atualizado, exigindo assim um maior processamento do computador. Na Figura 15 este processo é demonstrado com um diagrama.

Figura 15 - Game Loop



Fonte: Game Programming Patterns (2014)

O Pygame oferece diversas funções para auxiliar no desenvolvimento do Game Loop, abaixo algumas funções utilizadas no estudo de caso:

- *pygame.event.get()*, responsável pelo método *input,* essa função recebe todos os eventos enviados pelo jogador desde a última atualização.
- *pygame.sprite.Group.update()*, responsável pelo método *update*, essa função atualiza todos os *sprites* de um determinado grupo.
- *pygame.sprite.Group.draw(surface)*, responsável pelo método *render,* essa função mescla todos os *sprites* de um determinado grupo.

- *pygame.display.flip(),* responsável pelo método *render*, essa função atualiza a última superfície desenhada para a qual o método *pygame.sprite.Group.draw(surface)* desenhou nesse ciclo.
- *pygame.time.Clock.tick(FPS),* responsável pelo controle do tempo, esse método é executado uma vez por quadro, garantindo que o programa não seja processado mais rápido que o parâmetro informado.

4.5 VANTAGENS E DESVANTAGENS

Como apresentado nos capítulos anteriores, uma das maiores vantagens do Python e sua biblioteca Pygame é capacidade de implementarem em multiplataforma, isso permite desenvolver jogos para diversos sistemas operacionais sem grandes alterações no código fonte.

Essa flexibilidade garante que os desenvolvedores tenham como atenção principal a lógica do jogo, evitando preocupações com a compatibilidade entre diferentes sistemas operacionais.

Algo que deve ser levado em consideração por desenvolvedores independentes é que o Pygame é livre, ou seja, totalmente gratuito. Qualquer jogo criado utilizando a biblioteca pode ser vendido sem a necessidade da compra de uma licença comercial.

A quantidade de informações disponíveis *online* é também uma grande vantagem do Pygame. Por existir desde o ano 2000, muitas experiências já foram compartilhadas pelos usuários e desenvolvedores em fóruns de programação, além disso, existem diversos livros como por exemplo “Python, Pygame and the Raspberry Pi” de Kelly (2013) e o “Pygame for Python Game Development How-to: Create engaging and fun games with Pygame, Python’s Game development library” de Idris (2013), além de tutoriais e a própria documentação da biblioteca que apresenta o processo da programação de jogos utilizando Pygame.

A criação de jogos 3D e isométricos é possível no Pygame. Como apresentado no último capítulo, existem jogos profissionais desenvolvidos com Pygame, que não se limitam apenas ao 2D.

Pelo fato do Pygame ter sido lançado há um grande tempo, ele está sujeito a se tornar obsoleto de acordo com que novas tecnologias que surgem no mercado.

A medida com que os anos passam a evolução tecnológica cresce, e essas mudanças podem ser inviáveis para as atualizações de uma biblioteca com a idade do Pygame.

Na abordagem de desenvolvimento de jogos na atualidade, logo é mencionado as poderosas *engines*, que são *software* criados especificamente para a criação de *games.* Geig (2013) menciona em seu artigo sobre as grandes vantagens da *engine* Unity, que apresenta benefícios maiores comparado ao Pygame. Um grande exemplo desses benefícios, é a animação dos *sprites*, que no Pygame é feita via código e na *engine* Unity é feito por interface gráfica (UNITY TECHNOLOGIES, 2014a).

Porém tudo isto tem um custo, como já dito anteriormente, Pygame é totalmente gratuito o que permite desenvolver, publicar e vender um jogo totalmente livre de custos, no caso da utilização da engine Unity, para que um jogo possa ser comercializado é necessário uma licença Pro, que até a presente data pode ser adquirida por mil e quinhentos dolares o pacote com algumas restrições (UNITY TECHNOLOGIES, 2014b).

5 CONSIDERAÇÕES FINAIS

O projeto para o desenvolvimento de um jogo precisa ter primeiramente suas variáveis calculadas, deve-se analisar o público-alvo, o porte do jogo, o capital disponível aos desenvolvedores entre outros fatores, pois sabendo que existem diversas maneiras de se desenvolver um jogo, é necessário analisar qual delas é a melhor para o projeto, visando trazer o melhor custo-benefício.

O Pygame mostrou-se muito funcional quando utilizado no desenvolvimento de um jogo de pequeno porte. Com algumas horas de leitura da documentação da biblioteca, foi possível desenvolver um jogo de plataforma 2D com todos os recursos básicos de um jogo (gráficos, sons e mecânica). Entretanto, o Pygame mostrou-se também uma ferramenta com grande potencial para o desenvolvimento de jogos de grande porte, como apresentado ao longo do trabalho, jogos estes que foram desenvolvidos com o uso da biblioteca e obtiveram sucesso comercial.

Independente do capital disponível, o Pygame é sempre uma boa opção, inclusive para empresas já estabelecidas no ramo de desenvolvimento de jogos, por ser *open source*, ele pode ser adaptado pela própria empresa para se adequar as necessidades da mesma.

Outro ponto importante é em relação a distribuição quando o produto final estiver completo, poderá ser vendido sem nenhum custo adicional de licença. Além disso os desenvolvedores possuem um melhor controle do código, de forma que podem utiliza-lo para implementação de conceitos conhecidos, como o *Game Loop,* ou até mesmo para a criação de novos conceitos de desenvolvimento e controle do jogo.

Para trabalhos futuros indica-se o estudo de bibliotecas *open source* recentes do Python que permitam a criação de jogos multiplataforma, tais como a Kivy, Panda3D, Python-Ogre e Pyglet, com a intenção de apresentar aos desenvolvedores independentes novas maneiras para a criação de jogos utilizando ferramentas de código aberto.

GLOSSÁRIO

Char: Carácter. Pode ser um número, uma letra ou um símbolo.

Float: Número decimal.

Indentação: Recuo do texto em relação a margem.

Input: Entrada do usuário, pode ser dada apertando uma tecla, mexendo no mouse, etc.

Int: Número inteiro.

Multijogadores: Mais que um jogador.

Output: Resposta do computador, pode ser representado por uma imagem na tela, um som tocado, etc.

Palavra reservada: Palavra que não pode ser utilizada para nomear funções, variáveis, métodos, classes, etc. Todas as palavras reservadas têm um significado especial e diferente na linguagem de programação.

Print: Mostrar na tela.

Var: Palavra reservada do Javascript para a declaração de variáveis.

REFERÊNCIAS BIBLIOGRÁFICAS

2X2 GAMES. **Unity of Command**. Zagareb: 2x2 Games, 2012. Mídia Digital.

APPLE INC. **Mac OS X**. [S.l.]: Inc, Apple, 2001. CD.

BE INCORPORATED. **BeOS**. California: Be Incorporated, 1996. CD.

CASTLE TECHNOLOGY; RISC OS OPEN. **RISC OS**. Cambridge: RISC OS Open; Castle Technology, 1987.

CRAVEN, P. V. Program Arcade Games. **Program Arcade Games**, 2014. Disponivel em: <http://programarcadegames.com./index.php>. Acesso em: 4 Setembro 2014.

DESURA. Unity of Command Windows, Mac game | Desura. **Desura**, 4 Março 2012. Disponivel em: <http://www.desura.com/games/unity-of-command>. Acesso em: 21 Outubro 2014.

DIGITAL ILLUSIONS CE. **Battlefield 2**. Stockholm: EA Games, 2005. DVD, CD.

DIGITAL ILLUSIONS CE. **Battlefield 2142**. Stockholm: EA Games, 2006. DVD, Mídia Digital.

FRETS ON FIRE. About Frets On Fire. **Frets On Fire**, s.d. Disponivel em: <http://fretsonfire.sourceforge.net/about/>. Acesso em: 21 Outubro 2014.

GEIG, M. Why You Should Be Using the Unity Game Engine || InformIT. **Informit.com**, 2013. Disponivel em: <http://www.informit.com/articles/article.aspx?p=2031153>. Acesso em: 12 Novembro 2014.

HETLAND, M. L. **Beginning Python:** From Novice to Professional. 1ª. ed. New York: Apress, v. I, 2005.

HEWLETT-PACKARD. **HP-UX**. [S.l.]: Hewlett-Packard, 1984.

IBM. **AS/400**. [S.l.]: IBM, 1988.

IBM. **OS/390**. [S.l.]: IBM, 1995.

IBM; MICROSOFT CORPORATION. **OS/2**. [S.l.]: IBM, 1987. Disquete.

IDRIS, I. **Pygame for Python Game Development How-to:** Create engaging and fun games with Pygame, Python's Game development library. 1ª. ed. Birmingham: Packt Publishing Ltd., v. I, 2013.

KELLY, S. **Python, Pygame and the Raspberry Pi**. 1ª. ed. Ontario: [s.n.], v. I, 2013.

KYÖSTILÄ, S. E. A. **Freets On Fire**. [S.l.]: Unreal Voodoo, 2006. Mídia Digital.

MCKEOWN, J. S. **Programming in Visual Basic 2010:** the very beginner's guide. 1ª. ed. New York: Cambridge University Press, v. I, 2010.

MICROSOFT CORPORATION. **MS-DOS**. [S.l.]: Microsoft Corporation, 1981.

MICROSOFT CORPORATION. **Windows**. [S.l.]: [s.n.], 1985. Disquete.

MICROSOFT CORPORATION. **Windows CE**. [S.l.]: Microsoft Corporation, 1996. Mídia Digital.

NOKIA. **Symbian**. [S.l.]: Nokia, 1997. Mídia Digital.

NYSTROM, B. Game Loop - Sequencing Patterns - Game Programming Patterns. **Game Programming Patterns**, 2014. Disponivel em: <http://gameprogrammingpatterns.com/game-loop.html>. Acesso em: 29 Outubro 2014.

OPEN SOURCE INITIATIVE. The Open Source Initiative | Open Source Initiative. **Opensource.org**, 2014. Disponivel em: <http://opensource.org/>. Acesso em: 12 Novembro 2014.

OPENGL ORG. OpenGL Overview. **Opengl.org**. Disponivel em: <https://www.opengl.org/about/>. Acesso em: 13 Novembro 2014.

ORACLE CORPORATION. **Solaris**. [S.l.]: [s.n.], 1992.

PEMBERTON, S. The ABC Programming Language a short introduction. **Homepages**, 2012. Disponivel em: <http://homepages.cwi.nl/~steven/abc/>. Acesso em: 14 Novembro 2014.

PINHEIRO, F. D. A. C. **Elementos de Programação em C**. 1ª. ed. Porto Alegre: Bookman, v. I, 2012.

PYGAME ORG. Downloads. **Pygame.org**, Agosto 2009. Disponivel em: <http://pygame.org/download.shtml>. Acesso em: 21 Outubro 2014.

PYGAME ORG. Wiki. **Pygame.org**, 2014a. Disponivel em: <http://pygame.org/wiki/about>. Acesso em: 15 Outubro 2014.

PYGAME ORG. Pygame Documentation. **Pygame.org**, 2014b. Disponivel em: <http://www.pygame.org/docs/>. Acesso em: 26 Outubro 2014.

PYGAME ORG. pygame.Surface - Pygame v1.9.2 documentation. **Pygame.org**, 2014c. Disponivel em: <http://www.pygame.org/docs/ref/surface.html>. Acesso em: 1 Novembro 2014.

PYGAME ORG. pygame.sprite - Pygame v1.9.2 documentation. **Pygame.org**, 2014d. Disponivel em: <http://www.pygame.org/docs/ref/sprite.html>. Acesso em: 1 Novembro 2014.

PYGAME ORG. pygame.Rect - Pygame v1.9.2 documentation. **Pygame.org**, 2014e. Disponivel em: <http://www.pygame.org/docs/ref/rect.html>. Acesso em: 1 Novembro 2014.

PYTHON SOFTWARE FOUNDATION. General Python FAQ - Python 2.7.8 documentation. **Python Software Foundation | Python.org**, 2014a. Disponivel em: <https://docs.python.org/2/faq/general.html#what-is-the-python-software-foundation>. Acesso em: 4 Setembro 2014.

PYTHON SOFTWARE FOUNDATION. Download Python for Other Platforms | Python.org. **Python Software Foundation | Python.org**, 2014b. Disponivel em: <https://www.python.org/download/other/>. Acesso em: 23 Outubro 2014.

PYTHON SOFTWARE FOUNDATION. OrganizationsUsingPython - Python Wiki. **Python Software Foundation | Python.org**, 2014c. Disponivel em: <https://wiki.python.org/moin/OrganizationsUsingPython>. Acesso em: 23 Outubro 2014.

PYTHON SOFTWARE FOUNDATION. Python Documentation by Version | Python.org. **Python Software Foundation | Python.org**, 4 Outubro 2014d. Disponivel em: <https://www.python.org/doc/versions/>. Acesso em: 21 Outubro 2014.

PYTHON SOFTWARE FOUNDATION. Python2orPython3 - Python Wiki. **Python Software Foundation | Python.org**, 2014e. Disponivel em: <https://wiki.python.org/moin/Python2orPython3>. Acesso em: 18 Setembro 2014.

PYTHON SOFTWARE FOUNDATION. What's New in Python - Python 3.4.2 documentation. **Python Software Foundation | Python.org**, 2014f. Disponivel em: <https://docs.python.org/3/whatsnew/>. Acesso em: 18 Setembro 2014.

SHINNERS, P. Python Pygame Introduction. **Pygame Org**, s.d. Disponivel em: <http://www.pygame.org/docs/tut/intro/intro.html>. Acesso em: 26 Outubro 2014.

SIMPLE DIRECTMEDIA LAYER ORG. About SDL. **Libsdl.org**, 2014. Disponivel em: <https://www.libsdl.org/index.php>. Acesso em: 26 Outubro 2014.

SMITH, J. A. **Java Programs to Accompany Programming Logic and Design**. 6ª. ed. Boston: Cengage Learning, v. I, 2011.

STEAM. Acordo de Assinatura do Steam. **Store.Steampowered.com**, 2014. Disponivel em: <http://store.steampowered.com/subscriber_agreement/>. Acesso em: 12 Novembro 2014.

SWEIGART, A. **Making Games with Python & Pygame**. San Francisco: [s.n.], 2012. 347 p. Disponivel em: <http://inventwithpython.com/makinggames.pdf>. Acesso em: 4 Setembro 2014.

SWEIGART, A. What Professional Games Use Pygame? | The "Invent with Python" Blog. **The "Invent with Python" Blog | News about Al Sweigart's programming**

books, 19 Fevereiro 2013. Disponivel em: <http://inventwithpython.com/blog/2013/02/19/what-professional-games-use-pygame/>. Acesso em: 16 Outubro 2014.

THE MORPHOS DEVELOPMENT TEAM. **MorphOS**. [S.l.]: The MorphOS Development Team, 2000.

TIOBE SOFTWARE. TIOBE Software: Tiobe Index. **TIOBE Software**, Outubro 2014. Disponivel em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acesso em: 22 Outubro 2014.

UNITY TECHNOLOGIES. Unity - 2D Character Controllers. **Unity3d.com**, 2014a. Disponivel em: <https://unity3d.com/pt/learn/tutorials/modules/beginner/2d/2d-controllers>. Acesso em: 13 nov. 2014.

UNITY TECHNOLOGIES. Unity - Store. **Store.Unity3d.com**, 2014b. Disponivel em: <https://store.unity3d.com/pt-BR/products/pricing>. Acesso em: 13 nov. 2014.

W3SCHOOLS. JavaScript Variables. **W3Schools.com**, 2014. Disponivel em: <http://www.w3schools.com/js/js_variables.asp>. Acesso em: 12 Novembro 2014.

WEBZEN YMIR. **Metin2**. [S.l.]: OnGame, 2004. Mídia Digital.

WEBZEN YMIR GAMES. Webzen Ymir Games. **Webzen Ymir Games**, 2012. Disponivel em: <http://www.ymirgames.co.kr/game/metin2.htm#mn1>. Acesso em: 7 Novembro 2014.