Projeto Integrador
PyGame

Guilherme Brandão da Silva,
brandaogbs.github.io

PyGame! Introdução.

>>> **O que é?**

>>> Pra que serve?

>>> Exemplos

PyGame! Introdução.

>>> O que é?

>>> **Pra que serve?**

>>> Exemplos

_

PyGame! Introdução.

>>> O que é?

>>> Pra que serve?

>>> **Exemplos**

Cursos Online

PyGame Tutorial

https://www.youtube.com/watch?v=i6xMBig-pP4

Making a Platformer

https://www.youtube.com/watch?v=Qdeb1iinNtk

Building a Tetris

https://www.youtube.com/watch?v=zfvxp7PgQ6c

PyGame! Instalação.

>>> Instalar o **Python 3+**

>>> Instalar o **pip3**

>>> Instalar o **pygame**

python3 -m pip install -U pygame --user

>>> Instalar o **vscode**

PyGame! Primeiros Passos.

>>> **Importando o pygame**

>>> **Testando o pygame**

>>> Loop

PyGame!

Código !

PyGame! Primeiros Passos.

>>> Importando o pygame

>>> Testando o pygame

>>>Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!Loop!

PyGame!

Speed run!

PyGame! Eventos.

>>> **São gerados de forma assíncrona dentro da engine do jogo, os eventos são armazenados em uma queue, para capturar um evento basta utilizar o *event.get*().**

>>> Os eventos podem ser diversos tipos, como: botão pressionado, botão solto e mouse. Estes estão no dicionário *event.EventType*.

>>> Exemplo de eventos.

PyGame! Eventos.

>>> São gerados de forma assíncrona dentro da engine do jogo, os eventos são armazenados em uma queue, para capturar um evento basta utilizar o *event.get*().

>>> **Os eventos podem ser diversos tipos, como: botão pressionado, botão solto e mouse. Estes estão no dicionário *event.EventType*.**

>>> Exemplo de eventos.

PyGame! Eventos.

>>> São gerados de forma assíncrona dentro da engine do jogo, os eventos são armazenados em uma queue, para capturar um evento basta utilizar o *event.get*().

>>> Os eventos podem ser diversos tipos, como: botão pressionado, botão solto e mouse. Estes estão no dicionário *event.EventType*.

>>> **Exemplo de eventos.**

PyGame!

Código !

PyGame! Janela.

>>> A janela (*surface*) é onde a mágica acontece, todas as interações do jogo ocorrem dentro dela, como: visualização de objetos, colisões, atualização da posição do jogador e etc.

>>> Portanto, em cada iteração do loop deve ocorrer uma atualização (*update*) que resulta em frame novo da nossa janela.

PyGame! Janela.

>>> A janela (*surface*) é onde a mágica acontece, todas as interações do jogo ocorrem dentro dela, como: visualização de objetos, colisões, atualização da posição do jogador e etc.

>>> **Portanto, em cada iteração do loop deve ocorrer uma atualização *update()* que resulta em frame novo da nossa janela.**

PyGame! Janela.

>>> Além disso, podemos utilizar a função *tick()* para obter uma contagem fixa de frames.

>>> Legal, mas e essa tela preta?

PyGame! Janela.

>>> Além disso, podemos utilizar a função *tick()* para obter uma contagem fixa de frames.

>>> **Legal, mas e essa tela preta? Podemos pintar o fundo através da função *fill*().**

PyGame!

Código !

PyGame! Imagens.

>>> Alguns objetos no jogo podem ser pintados, tal como o fundo. Porém, pode ser interessante inserir imagens dentro do jogo, como por exemplos: o background, alguns objetos e o próprio player.

>>> Para conseguir colocar uma imagem dentro do carregamos ela através do *image.load*().

PyGame! Imagens.

>>> Alguns objetos no jogo podem ser pintados, tal como o fundo. Porém, pode ser interessante inserir imagens dentro do jogo, como por exemplos: o background, alguns objetos e o próprio player.

>>> **Para conseguir colocar uma imagem dentro do carregamos ela através do *image.load*().**

PyGame! Imagens.

>>> **Uma imagem parada nem sempre é interessante. Na realidade o que gostaríamos é de falar onde a imagem deve estar em cada frame.**

>>> Então, depois de carregada podemos atualizar o local da imagem através da função *blit*().

PyGame! Imagens.

>>> Uma imagem parada nem sempre é interessante. Na realidade o que gostaríamos é de falar onde a imagem deve estar em cada frame.

>>> **Então, depois de carregada podemos atualizar o local da imagem através da função *blit*().**

PyGame!

Código !

PyGame! Movimentos.

>>> **Já sabemos como posicionar uma imagem em lugares da tela. O que precisamos é em cada novo frame atualizar a posição da imagem, simulando assim o movimento.**

>>> Os movimentos podem ser de duas formas: ou são gerados pelo jogador (teclado) ou são elementos do jogo.

>>> Para o primeiro caso, devemos capturar os eventos e tratá-los dentro do loop.

PyGame! Movimentos.

>>> Já sabemos como posicionar uma imagem em lugares da tela. O que precisamos é em cada novo frame atualizar a posição da imagem, simulando assim o movimento.

>>> **Os movimentos podem ser de duas formas: ou são gerados pelo jogador (teclado) ou são elementos do jogo.**

>>> Para o primeiro caso, devemos capturar os eventos e tratá-los dentro do loop.

PyGame! Movimentos.

>>> Já sabemos como posicionar uma imagem em lugares da tela. O que precisamos é em cada novo frame atualizar a posição da imagem, simulando assim o movimento.

>>> Os movimentos podem ser de duas formas: ou são gerados pelo jogador (teclado) ou são elementos do jogo.

>>> **Para o primeiro caso, devemos capturar os eventos e tratá-los dentro do loop.**

PyGame! Movimentos.

>>> **Para cada um dos movimento, devemos atualizar o valor atual da posição conforme o desejado.**

>>> Lembre-se que a tela, nada mais é que uma matriz, portanto para movimentar a nossa personagem pelo eixo x devemos apenas somar ou subtrair do valor atual de x.

PyGame! Movimentos.

>>> Para cada um dos movimento, devemos atualizar o valor atual da posição conforme o desejado.

>>> **Lembre-se que a tela, nada mais é que uma matriz, portanto para movimentar a nossa personagem pelo eixo x devemos apenas somar ou subtrair do valor atual de x.**

PyGame!

Código !

PyGame! Bordas.

>>> Nossa personagem já consegue se movimentar pela tela e infelizmente até fora dela. Devemos agora refletir um pouco sobre como limitar as bordas do mapa e também sobre possíveis colisões que devem acontecer nos jogos.

>>> A forma mais simples de bloquear a movimentação de algo dentro do jogo é literalmente 'bloquear a movimentação'. Caso um objeto atinja uma região que não deve, podemos simplesmente zerar sua movimentação ou

PyGame! Bordas.

>>> Nossa personagem já consegue se movimentar pela tela e infelizmente até fora dela. Devemos agora refletir um pouco sobre como limitar as bordas do mapa e também sobre possíveis colisões que devem acontecer nos jogos.

>>> **A forma mais simples de bloquear a movimentação de algo dentro do jogo é literalmente 'bloquear a movimentação'. Caso um objeto atinja uma região que não deve, podemos simplesmente zerar sua movimentação ou mesmo mudar sua direção.**

PyGame!

Código !

PyGame! Obstáculos.

>>> Para tornar o jogo mais divertido, vamos precisar adicionar obstáculos e inimigos ao nosso jogo. Já vimos como criar personagens com imagens, mas uma ideia que também pode ser utilizada é a criação de formas geométricas.

>>> Os blocos podem ser criados através do método *draw.rect*().

PyGame! Obstáculos.

>>> Para tornar o jogo mais divertido, vamos precisar adicionar obstáculos e inimigos ao nosso jogo. Já vimos como criar personagens com imagens, mas uma ideia que também pode ser utilizada é a criação de formas geométricas.

>>> **Os blocos podem ser criados através do método *draw.rect*().**

PyGame! Obstáculos.

>>> A ideia é criar obstáculos que entrem em rota de colisão com a nossa personagem. Portanto, vamos fazer com que blocos aleatórios caiam no mapa e nossa personagem tenha que desviar deles.

>>> Os blocos devem começar na posição 0 (topo do mapa) e caírem até a parte mais baixa.

PyGame! Obstáculos.

>>> A ideia é criar obstáculos que entrem em rota de colisão com a nossa personagem. Portanto, vamos fazer com que blocos aleatórios caiam no mapa e nossa personagem tenha que desviar deles.

>>> **Os blocos devem começar na posição 0 (topo do mapa) e caírem até a parte mais baixa.**

PyGame!

Código !

PyGame! Colisão.

>>> **Assim como realizamos a limitação do mapa através de bordas, podemos realizar a limitação de dois objetos.**

>>> A ideia permanece a mesma, devemos encontrar as bordas do objeto e então verificar se um objeto está em contato com o outro.

PyGame! Colisão.

>>> Assim como realizamos a limitação do mapa através de bordas, podemos realizar a limitação de dois objetos.

>>> A **ideia permanece a mesma, devemos encontrar as bordas do objeto e então verificar se um objeto está em contato com o outro.**

PyGame!

Código !

PyGame! Texto.

>>> Para deixar o jogo mais interativo, em alguns momentos talvez seja interessante exibir mensagens para o jogador, como início do jogo, fim do jogo e etc.

PyGame!

Código !

PyGame! Scores.

>>> O jogo está quase completo, falta só apresentar as pontuações, para que o jogador consiga visualizar em qual fase ele está ou quantos inimigos ele eliminou e etc.

>>> Além disso, pode ser interessante aumentar a dificuldade conforme a pontuação também aumenta.

PyGame! Scores.

>>> O jogo está quase completo, falta só apresentar as pontuações, para que o jogador consiga visualizar em qual fase ele está ou quantos inimigos ele eliminou e etc.

>>> **Além disso, pode ser interessante aumentar a dificuldade conforme a pontuação também aumenta.**

PyGame!

Código !