Na plataforma do replit já podemos criar um projeto que contenha o Pygame, então basta criar um novo projeto e no lugar da linguagem selecionar o **Pygame.**

Graphical user interface, text, application, chat or text message

Description automatically generated

Com o projeto criado já podemos trabalhar com o Pygame, e o primeiro passo é fazer a importação do Pygame e também ja vamos fazer a inicialização dele

Text

Description automatically generated

Com esse código acima teremos o seguinte resultado:

Graphical user interface, text

Description automatically generated

Agora temos que definir uma forma de saber se nosso jogo continua rodando ou se ele foi finalizado

Text

Description automatically generated

Nosso jogo já está começando a tomar forma, e para nos próximos conteúdos veremos como desenhar objetos na nossa tela e também para fazer movimentação deles na tela.

## Documentação do Pygame

A documentação do Pygame está toda em inglês, mas caso tenha interesse e verificar o funcionamento acessa a documentação dele [aqui](https://www.pygame.org/docs/).

## Desenhando Objetos

O Pygame já vem com uma opção para podemos fazer alguns desenhos básicos como retângulos, círculos, polígonos e linhas. E estaremos utilizando essa opção para não precisarmos adicionar imagens e fazer a utilização dela.

Então vamos ver como criar algumas dessas formas:

Text

Description automatically generated

Ao executar o seguinte código, vemos que na tela o retângulo e o círculo que nós criamos:

Graphical user interface, application

Description automatically generated

Agora vamos entrar em detalhes do que nosso código está fazendo

Primeiro vamos dar uma olhada nas nossas variáveis **red** e **green**, a tupla que usamos para definir o valor dessa variável representa o **RGB** da cor. Caso queira selecionar uma cor diferente basta pesquisar na internet por **Color picker** e terá algumas alternativas, agora é só escolher a cor deseja e copiar os valores do **RGB** e pronto.

Agora a função **pygame.display.update()** faz a atualização da nossa tela, isso quer dizer que vai ser exibido tudo que foi adicionado ao nosso display.

A função **pygame.draw.rect(dis, red, [position\_x, position\_y, square\_size, square\_size])** recebe como parâmetros o display que criamos, uma cor e uma lista com os valores com o seu formato. Os dois primeiros valores desse formato(**postion\_x e position\_y**) indicam a esquerda e o topo do retângulo, isso quer dizer que o tamanho desse retângulo será feito a direito e abaixo desse ponto. E por último os dois últimos valores indicam o **width**(largura) e **height**(altura) que no nosso caso como foi um quadrado utilizamos o mesmo valor para ambos.

Por último a função **pygame.draw.circle(dis, green, (position\_x + 200, position\_y), circle\_radius)** é bem semelhante ao do retângulo, os dois primeiros parâmetros tem a mesma funcionalidade. O terceiro parâmetro é um tupla que indica qual vai ser o centro do círculo, por isso na imagem ele ficou um pouco acima do quadrado. E o último parâmetro é o raio desse círculo, caso não lembre o que seja raio é a distância de um ponto da borda do círculo até o centro.

## Atenção ao Posicionamento

Acabamos de ver como faz para desenhar um objeto no Pygame, mas precisamos prestar atenção com os valores do posicionamento.

O ponto (0, 0) fica no canto superior esquerdo, portanto o ponto (dis\_width, dis\_height) fica no canto inferior direito. Dessa forma se desenharmos um círculo nesses dois pontos teremos o seguinte:

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Note que apenas uma parte do círculo está aparecendo, pois, o centro dele se encontra nesses extremos da tela.

É importante entendermos essa questão do posicionamento para quando fomos fazer a movimentação desses desenhos.

Agora que já sabemos como desenhar objetos na nossa tela, mas eles estão parados e em um jogo normalmente temos objetos se movimentando. Portanto, precisamos fazer esse objeto que foi desenhado se movimentar pela nossa tela, para isso precisamos alterar a posição dele a cada vez que entrarmos no loop da nossa tela.

Text

Description automatically generated

Nesse código utilizamos o **clock.tick(speed)**, o motivo é para limitar os frames por segundo máximo na máquina, isso quer dizer que se o computador que estiver executando for muito potente o jogo funcione da mesma forma.

Agora vamos executar para ver o que o resultado desse código:

Graphical user interface, text

Description automatically generated

Mas o que aconteceu nessa aplicação? Porque ficou essa linha vermelha?

O motivo dessa linha ficar aparecendo é bem simples, a cada vez que executamos o nosso loop nós desenhamos um novo quadrado sem remover o anterior. Então o que temos são diversos quadrados que foram desenhados em posições diferentes.

A forma de resolver isso é bem simples, apenas precisamos fazer algumas alterações:

Text

Description automatically generated

Esse comando **dis.fill()** vai substituir tudo que está presente na tela pela cor que for passada, dessa forma vai remover o quadrado que foi desenhado anteriormente.

Agora ao executar teremos o seguinte:

Graphical user interface

Description automatically generated with medium confidence

Agora podemos ver que é exibido apenas um quadrado pela tela, mas para onde ele foi?

Como não fazemos nenhuma verificação ele continuou nessa direção, se você fazer o **print** da variável **position\_y** você consegue ver que ela vai aumentando indefinidamente. E para evitar que isso aconteça precisamos colocar uma trava para indicar qual o valor máximo de y que nosso desenho pode ir. E para aproveitar já vamos definir também o valor minimo e fazer o mesmo para o x.

Precisamos alterar nosso código da seguinte forma para corrigir esse problema:

Text

Description automatically generated

Executando agora teremos o seguinte:

Graphical user interface, text, application

Description automatically generated

Pronto! Agora nosso quadrado não está indo mais para o infinito e além.

Para a movimentação seria isso, para que o quadrado mova para "cima" precisaríamos subtrair o **speed** do **position\_y**, enquanto para mover para esquerda e direita precisaríamos alterar o **position\_x**.

Já vimos como faz para movimentar um desenho pela nossa tela, mas ainda está faltando adicionar uma forma de controlar esse movimento. Então é isso que iremos fazer, vamos partir do último código que fizemos sobre movimentação e fazer algumas alterações para ser possível controlar a movimentação.

Text

Description automatically generated

Quando executarmos esse código teremos o seguinte resultado:

Graphical user interface, application

Description automatically generated

Pela imagem não da para perceber, mas ao pressionarmos qualquer tecla de movimentação o nosso bloco irá se mover naquela direção indefinidamente. E esse pode ser um comportamento que não queremos e para isso precisaremos fazer uma pequena alteração:

Text

Description automatically generated

Dessa forma teremos o seguinte resultado:

Graphical user interface, application

Description automatically generated

Agora esbarramos em outro problema, ao executar e mantermos a tecla pressionada só ira efetuar o movimento uma vez. Precisamos alterar esse comportamento para que enquanto estivermos pressionando a tecla continue o movimento.

Alterando novamente o loop, teremos o seguinte:

Text

Description automatically generated

Agora nossa movimentação ficará da seguinte forma:

Graphical user interface, application

Description automatically generated

## Utilizando Outras Teclas

Nessa movimentação, nós utilizamos as setas do teclado para controlar ela, mas poderíamos ter utilizado qualquer outra tecla para isso. Por exemplo, podemos permitir a utilização do **WASD** para controlar a movimentação além das setas, para isso precisamos alterar os nossos **ifs** das teclas pressionadas da seguinte forma:

E fazer algo parecido para os outros **ifs**, apenas trocando a tecla que vai ser verificada. Lembrando de adicionar a tecla no evento **KEYUP**.

Em algumas situações podemos querer adicionar textos ao nosso jogo, como nós casos que nós queremos exibir a pontuação do jogador. E para fazer isso é bastante simples de se fazer, e o primeiro passo seria cria uma fonte:

Graphical user interface, text

Description automatically generated

Nesse código buscamos uma fonte que esteja presente no sistema, o primeiro parâmetro é o nome da fonte e o segundo parâmetro é o tamanho.

Agora vamos criar uma função que vai ser responsável para exibir esse nosso texto:

Text

Description automatically generated

Antes de continuar vamos dar uma analisada nessa função, primeiro vamos para essa função **render**, ela recebe como parâmetro o texto que será utilizado na renderização e pode ser qualquer string, o segundo parâmetro é um booleano para indicar se utiliza antialias e isso serve para tirar o serrilhado do texto e por último o terceiro parâmetro é a cor do texto a ser renderizado.

Agora essa função **blit**, serve para colocar uma imagem sobre a outra, no caso estamos colocando a imagem do texto sobre o fundo. O segundo parâmetro é para indicar o posicionamento dessa imagem.

O próximo passo é fazer a chamada dessa função:

Text

Description automatically generated

Após executar o nosso código teremos algo parecido com o seguinte resultado:

Graphical user interface

Description automatically generated with medium confidence

É isso para esse conteúdo, ele é um pouco mais simples. Mas lhe permite que sejam adicionados diversos conteúdos de texto ao seu jogo.

Já aprendemos como adicionar desenhos na nossa tela e fazer a movimentação e isso pode ser o suficiente para começar a fazer o protótipo de um jogo.

Mas para publicar um jogo vamos querer adicionar imagens próprias e fazer isso é bem simples no Pygame, pois já possui uma forma de carregar imagens de sua máquina para o Pygame. E o primeiro passo para isso é carregar a imagem:

Text

Description automatically generated

Nesse trecho apenas carregamos nossa imagem e obtemos o retângulo que representa ela, por exemplo, de a imagem for 32x32 o nosso retângulo vai ser 32x32.

Para a movimentação vamos utilizar algo um pouco diferente do que fizemos com o quadrado, como temos o retângulo referente ao nosso player, vamos utilizar uma função ali que permite fazer essa movimentação. Assim nosso loop ficaria dessa forma:

Text

Description automatically generated

Esse código vai deixar uma movimentação automática para nosso player, igual à primeira movimentação que fizemos para o quadrado. Para adicionar o controle, apenas precisamos alterar o speed quando uma tecla for pressionada.

Agora vamos analisar o esse código fez, a primeira coisa é que definimos o **speed** e o **char\_speed** do personagem. A função **move\_ip** faz a movimentação do **player\_rect** na tela no eixo x e y utilizando a lista do **char\_speed**. E por último a função **blit** coloca a imagem do player na posição do **player\_rect.**

Com esse código teremos o seguinte resultado:

A picture containing graphical user interface

Description automatically generated

## Transformações na Imagem

Pode acontecer que seja necessário redimensionar a imagem ou até mesmo rotacionar ela para ficar conforme a necessidade que tem no jogo. Para isso temos algumas funções do Pygame para lidar com essas situações, e aqui estão algumas dessas funções:

* **pygame.transform.flip():** Inverte a imagem horizontalmente ou verticalmente
* **pygame.transform.scale():** Redimensiona a imagem
* **pygame.transform.rotate():** Rotaciona a imagem

Existem outras funções para fazer essas transformações, mas essas são uma das principais.