Matrizes para criação de 3 tipos de inimigos, uma nave e vidas, respectivamente:

“inimigo01.txt”

-1 -1 4 4 4 -1 -1

-1 -1 0 4 0 -1 -1

-1 -1 4 4 4 -1 -1

-1 4 -1 4 -1 4 -1

-1 4 -1 4 -1 4 -1

4 -1 -1 -1 -1 -1 4

-1 -1 -1 -1 -1 -1 -1

“inimigo02.txt”

-1 -1 -1 -1 -1 -1 -1

-1 -1 4 4 4 -1 -1

-1 -1 0 4 0 -1 -1

-1 -1 4 4 4 -1 -1

-1 4 4 4 4 4 -1

-1 4 -1 -1 -1 4 -1

4 -1 4 -1 4 -1 4

“inimigo03.txt”

-1 -1 -1 -1 -1 -1 -1

-1 -1 4 4 4 -1 -1

4 -1 0 4 0 -1 4

4 4 4 4 4 4 4

-1 -1 4 -1 4 -1 -1

-1 4 -1 4 -1 4 -1

-1 -1 -1 -1 -1 -1 -1

“MatrizExemplo0.txt

-1 4 4 -1 4 4 -1

4 4 4 4 4 4 4

4 4 4 4 4 4 4

-1 4 4 4 4 4 -1

-1 -1 4 4 4 -1 -1

-1 -1 -1 4 -1 -1 -1

“heart.txt”

-1 -1 -1 10 -1 -1 -1

-1 -1 -1 10 -1 -1 -1

-1 -1 10 0 10 -1 -1

-1 -1 10 2 10 -1 -1

-1 10 10 10 10 10 -1

-1 10 -1 10 -1 10 -1

“TransformacoesGeometricas.py”

Variáveis globais criadas:

tiros\_disparados = 0

tempo\_inicio = time.time()

limite\_tiros = 10

intervalo = 2

vidas = 3

intervaloCriacao= 2

proximo\_tiro\_inimigo = time.time() + 2

intervalo\_tiro\_inimigo = 4

Todos os modelos das matrizes são criados aqui:

    Modelos.append(ModeloMatricial())

    Modelos[0].leModelo("MatrizExemplo0.txt");

    Modelos.append(ModeloMatricial())

    Modelos[1].leModelo("MatrizProjetil.txt");

    Modelos.append(ModeloMatricial())

    Modelos[2].leModelo("inimigo02.txt");

    Modelos.append(ModeloMatricial())

    Modelos[3].leModelo("inimigo01.txt");

    Modelos.append(ModeloMatricial())

    Modelos[4].leModelo("inimigo03.txt");

    Modelos.append(ModeloMatricial())

    Modelos[5].leModelo("heart.txt")

Adicionado a possibilidade de acelerar e desacelerar com as setas:

def arrow\_keys(a\_keys: int, x: int, y: int):

    if a\_keys == GLUT\_KEY\_UP:         # Se pressionar UP

        Personagens[0].Velocidade=6

if a\_keys == GLUT\_KEY\_DOWN:       # Se pressionar DOWN

        Personagens[0].Velocidade=2

Em display são adicionadas as linhas:

def display():

criacaoDeInimigos()

    inimigosAtiram()

    DesenhaCoracao()

Para verificar impactos e atualizar os personagens foi adicionado em AtualizaPersonagens:

def AtualizaPersonagens(tempoDecorrido):

    if getattr(Personagens[i], "tipo", None) == "inimigo":

            VerificaColisaoComLimites(Personagens[i])

Para desenhar as vidas restantes do personagem foi adicionado o seguinte método:

def DesenhaCoracao():

    global vidas

    if vidas <= 0:

        return

    glPushMatrix()

    glLoadIdentity()  # desenha sempre em coordenadas fixas

    # posição inicial (canto superior esquerdo)

    margem\_x = Min.x + 2

    topo\_y = Max.y - 8  # 6 de altura + 2 de margem

    for i in range(vidas):

        glPushMatrix()

        glTranslatef(margem\_x + i \* 8, topo\_y, 0)  # cada coração tem 7 de largura + 1 espaço

        DesenhaModelo(5)  # índice do coração no Modelos

        glPopMatrix()

    glPopMatrix()

A partir de “DesenhaPersonagemMatricial.py” foi criado para poder executa-lo de outra forma o “DesenhaModelo”

def DesenhaModelo(idx):

    MM = Modelos[idx]

    larg = MM.nColunas

    alt = MM.nLinhas

    glPushMatrix()

    for i in range(alt):

        glPushMatrix()

        for j in range(larg):

            cor = MM.getColor(alt - 1 - i, j)

            if cor != -1:

                SetColor(cor)

                DesenhaCelula()

                SetColor(Wheat)

                DesenhaBorda()

            glTranslatef(1, 0, 0)

        glPopMatrix()

        glTranslatef(0, 1, 0)

    glPopMatrix()

Usado por exemplo para criar o coração das vidas

O método atualiza jogo ficou desta forma:

def AtualizaJogo():

    global imprimeEnvelope, nInstancias, Personagens, vidas

    # Atualiza envelopes de colisão de todos os personagens

    for i in range(0, nInstancias):

        AtualizaEnvelope(i)

        if imprimeEnvelope:

            #print("Envelope ", i)

            Personagens[i].ImprimeEnvelope("", "")

    imprimeEnvelope = False

    # Verifica colisao do jogador com inimigos ou tiros

    for i in range(1, nInstancias):

        personagem = Personagens[i]

        # Ignora com o do jogador

        if getattr(personagem, "tipo", None) == "tiro" and getattr(personagem, "atirador", None) == 0:

            continue

        if TestaColisao(0, i):  # jogador é o índice 0

            #print(f"Colisão com personagem {i} do tipo {personagem.tipo}")

            vidas -= 1

            #print(f"Vidas restantes: {vidas}")

            if vidas <= 0:

                print("FIM DE JOGO")

                keyboard(ESCAPE, 0, 0)  # Fim

            # Reposiciona o personagem dps da colisão

            Personagens[i] = copy.deepcopy(Personagens[i + AREA\_DE\_BACKUP])

            Personagens[i].Posicao = GeraPosicaoAleatoria()

            ang = random.randint(0, 360)

            Personagens[i].Rotacao = ang

            Personagens[i].Direcao = Ponto(0, 1)

            Personagens[i].Direcao.rotacionaZ(ang)

    # Verifica colisões de tiros do jogador com inimigos

    for i in range(1, nInstancias):

        if getattr(Personagens[i], "tipo", None) != "tiro":

            continue

        if getattr(Personagens[i], "atirador", None) != 0:

            continue  # Só tiros do jogador

        for j in range(1, nInstancias):

            if i == j:

                continue

            if getattr(Personagens[j], "tipo", None) != "inimigo":

                continue

            if TestaColisao(i, j):

                #print(f"Inimigo {j} foi atingido pelo tiro {i}!")

                # Reposiciona inimigo após a morte

                Personagens[j] = copy.deepcopy(Personagens[j + AREA\_DE\_BACKUP])

                Personagens[j].Posicao = GeraPosicaoAleatoria()

                ang = random.randint(0, 360)

                Personagens[j].Rotacao = ang

                Personagens[j].Direcao = Ponto(0, 1)

                Personagens[j].Direcao.rotacionaZ(ang)

                # move o tiro pra fora e para de movimentar

                Personagens[i].Velocidade = 0

                Personagens[i].Posicao = Ponto(9999, 9999)

                break  # Um tiro mata o inimigo

Para a criação de instâncias de personagens no jogo:

def CriaInstancias(TipoPersonagem: int):

    global Personagens, nInstancias

    i = TipoPersonagem

    ang = -90.0

    if TipoPersonagem == 0:  # Jogador da MatrizExemplo0, nave

        Personagens[i].Posicao = Ponto(-2.5, 0)

        Personagens[i].Escala = Ponto(1, 1)

        Personagens[i].Rotacao = ang

        Personagens[i].IdDoModelo = 0

        Personagens[i].Modelo = DesenhaPersonagemMatricial

        Personagens[i].Pivot = Ponto(3.5, 0)

        Personagens[i].Direcao = Ponto(0, 1)

        Personagens[i].Direcao.rotacionaZ(ang)

        Personagens[i].Velocidade = 3

        Personagens[i].tipo = "jogador"  # Tipo de jogador

        Personagens[i + AREA\_DE\_BACKUP] = copy.deepcopy(Personagens[i])

        nInstancias = 1  # Inicia com 1

    else:

        Personagens[i + AREA\_DE\_BACKUP] = copy.deepcopy(Personagens[i])

        nInstancias += 1

        ang = random.randint(0, 90)

        if TipoPersonagem == 1:  # exemplo de tiro

            Personagens[nInstancias].Posicao = Ponto(13.5, 0)

            Personagens[nInstancias].IdDoModelo = 1

        elif TipoPersonagem == 2:

            Personagens[nInstancias].Posicao = GeraPosicaoAleatoria()

            Personagens[nInstancias].IdDoModelo = 2

        elif TipoPersonagem == 3:

            Personagens[nInstancias].Posicao = GeraPosicaoAleatoria()

            Personagens[nInstancias].IdDoModelo = 3

        elif TipoPersonagem == 4:

            Personagens[nInstancias].Posicao = GeraPosicaoAleatoria()

            Personagens[nInstancias].IdDoModelo = 3

        Personagens[nInstancias].Escala = Ponto(1, 1)

        Personagens[nInstancias].Rotacao = ang

        Personagens[nInstancias].Modelo = DesenhaPersonagemMatricial

        Personagens[nInstancias].Pivot = Ponto(3.5, 0)

        Personagens[nInstancias].Direcao = Ponto(0, 1)

        Personagens[nInstancias].Direcao.rotacionaZ(ang)

        Personagens[nInstancias].Velocidade = 3

        Personagens[nInstancias].tipo = "inimigo"

        Personagens[nInstancias + AREA\_DE\_BACKUP] = copy.deepcopy(Personagens[i])

        nInstancias += 1

Para a mudança da cor do fundo de tela:

def init():

glClearColor(0,0,0,0)

Para a criação de inimigos:

def criacaoDeInimigos():

    global TempoInicial

    tempo\_decorrido = time.time() - TempoInicial

    if tempo\_decorrido >= random.randint(3,7):

        nroInimigo=random.randrange(2,4)

        CriaInstancias(nroInimigo)

        #reseta

        TempoInicial = time.time()

Função que avalia o tempo para atirar:

def pode\_atirar():

    global tiros\_disparados, tempo\_inicio

    agora = time.time()

    # se passou reseta o contador

    if agora - tempo\_inicio > intervalo:

        tiros\_disparados = 0

        tempo\_inicio = agora

    # Se ainda pode atirar

    if tiros\_disparados < limite\_tiros:

        tiros\_disparados += 1

        return True

    else:

        return False

Função que cria o tiro

def CriaTiro2(nAtirador: int):

    if not pode\_atirar():

        #print("Limite de tiros atingido. Aguarde 2 segundos")

        return

    global nInstancias, Personagens, Modelos, AREA\_DE\_BACKUP, DesenhaPersonagemMatricial

    i = nInstancias

    Atirador = Personagens[nAtirador]

    ang = Atirador.Rotacao

    Personagens[i].Escala = Ponto(1, 1)

    Personagens[i].Rotacao = ang

    Personagens[i].IdDoModelo = 1

    Personagens[i].Modelo = DesenhaPersonagemMatricial

    Personagens[i].Pivot = Ponto(0.5, 0)

    Personagens[i].Direcao = Ponto(0, 1)

    Personagens[i].Direcao.rotacionaZ(ang)

    envelope = Atirador.Envelope

    direcao = Personagens[i].Direcao

    pos\_base = Atirador.Posicao

    # Encontra os dois pontos mais na frente do envelope na direção do disparo

    frente = sorted(envelope, key=lambda p: (p - pos\_base).x \* direcao.x + (p - pos\_base).y \* direcao.y, reverse=True)[:2]

    centro\_frente = (frente[0] + frente[1]) \* 0.5

    # Posiciona o tiro um pouco na frente da ponta do atirador

    Personagens[i].Posicao = centro\_frente + direcao \* 0.3 - Personagens[i].Pivot

    Personagens[i].Velocidade = 5

    # marca como tiro e define quem atirou

    Personagens[i].tipo = "tiro"

    Personagens[i].atirador = nAtirador

    # backup

    Personagens[i + AREA\_DE\_BACKUP] = copy.deepcopy(Personagens[i])

    nInstancias += 1

Função que permite os inimigos atirarem com contagem de tempo:

def inimigosAtiram():

    global proximo\_tiro\_inimigo, intervalo\_tiro\_inimigo, nInstancias

    tempo\_atual = time.time()

    if tempo\_atual < proximo\_tiro\_inimigo:

        return

    for i in range(1, nInstancias):

        if getattr(Personagens[i], "tipo", None) == "inimigo":

            CriaTiro2(i)  # Inimigo atira

    # Define o próximo tempo de disparo

    proximo\_tiro\_inimigo = tempo\_atual + intervalo\_tiro\_inimigo

Métodos que avalia e devolve o inimigo a janela caso tenha tentado sair dos limites:

def VerificaColisaoComLimites(personagem):

    global Min, Max

    bateu = False

    # Checa colisão horizontal

    if personagem.Posicao.x < Min.x or personagem.Posicao.x > Max.x:

        personagem.Direcao.x \*= -1

        personagem.Posicao.x = max(Min.x, min(personagem.Posicao.x, Max.x))

        bateu = True

    # Checa colisão vertical

    if personagem.Posicao.y < Min.y or personagem.Posicao.y > Max.y:

        personagem.Direcao.y \*= -1

        personagem.Posicao.y = max(Min.y, min(personagem.Posicao.y, Max.y))

        bateu = True

    # Atualiza a rotação do gráfico se houve colisão

    if bateu:

        angulo = math.degrees(math.atan2(personagem.Direcao.y, personagem.Direcao.x))

        personagem.Rotacao = angulo - 90