
River Raid - 3ª fase

Junho de 2014

Alunos:

Bruno Sesso - 8536002

Gustavo Estrela de Matos - 8526051

Nikolai Jose Eustatios Kotsifas – 8536072

Primeiras decisões

No primeiro encontro que o grupo realizou foram feitas pequenas decisões que seriam necessárias para dar o início a codificação. As mais importantes foram:

- Por motivos de padronização, e maior facilidade de leitura do código, decidimos que o código seria escrito em inglês e os comentários seriam feitos em português.
- Criamos uma variável que indicasse a mira da nave, assim o jogador poderia atirar na direção que quisesse sem precisar mudar a direção da nave.
- Definições de structs ficariam nos arquivos .h
- O inimigos são estáticos, só atiram.
- Criação dos arquivos queue.h e queue.c para o armazenamento dos inimigos e projéteis da tela.
- Criação dos arquivos utils.h e utils.c com funções e variáveis úteis a mais de uma função.
- Usamos git para o controle de versões.
- Definimos como seriam os eixos (vide utils).

Segunda Fase

Nosso objetivo na segunda fase foi fazer o jogo jogável, porém sem parte gráfica. Portanto, o protagonista dessa fase foi o arquivo “main.c”. Nos demais arquivos, adicionamos algumas funções que viram a ser úteis.

Já com experiência obtida da primeira fase, fizemos bom uso do programa git e do repositório github para controlar e publicar versões novas, visto que não tivemos reuniões em um lugar físico, isto é, cada um fez sua parte do programa em lugares diferentes. Também baseado na experiência passada, decidimos determinar uma pessoa para organizar o que as demais faziam, para evitar código repetido, diferenças de padrões em nomes de variáveis, etc. Essa pessoa, então, é encarregada de juntar o que as demais fazem e projetar como usa-las em conjunto.

Terceira Fase

Fazer o jogo se tornar visível e jogável foi um trabalho árduo de aprendizagem de OpenGL. A falta de familiaridade com a biblioteca tomou bastante tempo de todos os membros do grupo.

Por haver partes gráficas e ser algo novo para todos os integrantes, houveram mais reuniões presenciais que nas demais fases. Continuando com o mesmo esquema feito na fase anterior, uma pessoa foi mais responsável em juntar os códigos dos demais. Assim, foi possível dividir o trabalho para cada pessoa fazer. Ao final cada um tinha trabalhado igualmente e ninguém se sentiu prejudicado.

A princípio, havia o interesse em colocar som no jogo. Mas devido às inconveniências de final de semestre não conseguimos realizar esse ponto.

Para construir toda a parte gráfica, não tivemos muitos problemas com as funções que já haviam sido implementadas. Fizemos uso da maioria delas e tivemos que fazer apenas pequenas alterações para adaptar a essa fase. Como as funções já estavam implementadas, foi possível se focar nas novas funções do OpenGL sem se preocupar muito com a dinâmica do jogo em si, isso é, o programa da fase anterior nos proporcionou uma boa base.

Decidimos nessa fase fazer com que cada elemento tivesse um raio que delimitava seu espaço. Isso não interferia na dinâmica do jogo e permitiu que houvesse modelos 3D sem se preocupar com cada ponto do objeto.

Optamos por não ter gravidade, pois deixa a jogabilidade melhor. Ainda assim implementamos a gravidade mas decidimos por deixa-la valendo 0.

Principais dificuldades

- Filtrar as diversas funções do OpenGL para aquilo que necessitávamos.
- Entender as matrizes que o OpenGL usa.
- Encontrar o vetor velocidade do tiro com apenas as coordenadas da tela.
- Redimensionar a janela mantendo a proporção.
- Colocar texturas nos modelos.

Como jogar?

Comando	Função
W A S D	Movimenta a nave
Mouse	Mira e atira
ESPAÇO	Aumenta a velocidade da nave
P	Diminui a velocidade da nave
M	Ativa o slow-motion

Estrutura do programa

Utils

Declaramos variáveis globais, structs, fazemos defines e funções de âmbito geral.

Cenario

Representa o espaço onde o jogo ocorre. Responsável por gerenciar os elementos que estão nele (verificar colisões e administrar os inimigos a serem criados e eliminados).

Ship

Representa a nave (jogador) que pode se mover, atirar, ser acertado, etc.

Enemy

Representam os inimigos que atiram na nave. Tem funções para os inimigos atirarem, serem atingidos, etc.

EnemyQueue

Representa uma fila onde cada elemento é um inimigo.

Shot

Representa um tiro podendo ser de um inimigo ou de uma nave.

ShotQueue

Representa uma fila onde cada elemento é um tiro.

Main

Cria todos os elementos, e executa o loop do jogo.

Hud

Desenha elementos independentes do cenário na tela, como o velocímetro, vida, etc.

Configuration

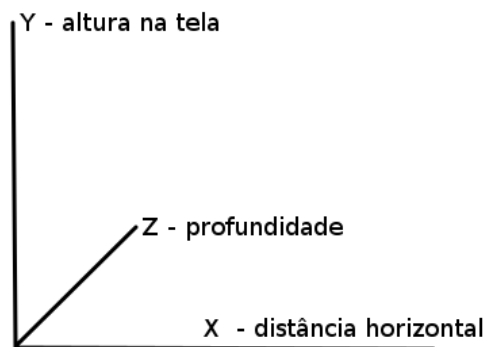
Contém constantes de configuração.

Descrição dos blocos

Utils

Arquivos: "utils.h", "utils.c".

O bloco Utils guarda os tipos Position, Velocity e Dimension, que são da struct coordinates. Essa struct tem como atributos as coordenadas x,y e z. As coordenadas x,y e z foram definidas assim:



Funções	
spaceTimeEquation()	atualiza uma Position de acordo com sua Velocity
distance()	calcula a distancia entre duas Position
speedTimeEquation()	Calcula uma nova Velocity devido à gravidade e a retorna
createPosition()	Cria uma estrutura do tipo Position
loadTexture()	Carrega um arquivo do tipo PPM ou PBM

Variáveis Globais	
clockTick	valor em segundos do tick do relógio
shipPosition	Posição em Z da nave
gameState	Indica se o jogo está em execução ou na tela inicial ou em game over

Cenário

Arquivos: “cenario.h”, “cenario.c”.

Atributos	
enemies	um ponteiro para uma fila implementada em uma lista ligada circular com cabeça
dimension	do tipo Dimension, indica o tamanho do cenário.

Funções	
createCenario()	cria e inicia o cenário e a fila enemies
refreshCenario()	atualiza a lista enemies de acordo com a posição da nave
createNewEnemyInInterval()	cria um inimigo num intervalo de forma aleatória
initEnemies()	inicia a lista de inimigos preenchendo o tamanho do buffer
verifyShipColision()	verifica se a nave colidiu com algum inimigo
verifyShipShotColision()	verifica se o tiro da nave atingiu algum inimigo
verifyEnemiesShotColision()	verifica se o tiro dos inimigos atingiu a nave
rmFarShots()	elimina tiros que estão muito distantes
freeCenario()	libera a memória ocupada pelo cenário
renderBackground()	desenha a imagem de fundo
renderCenario()	desenha os elementos do cenário

Ship

Arquivos: "ship.h", "ship.c".

Atributos	
life	um inteiro de 0 a 100 que diz o quanto a nave tem de "vida"
position	do tipo Position (vide utils), controla a posição da nave no espaço
velocity	do tipo Velocity (vide utils), controla a orientação e velocidade da nave no espaço
radius	raio da nave

Funções	
createShip()	cria a nave
killShip()	libera o espaço da memória usado pela nave
updateVelocity()	atualiza a velocidade da nave em função da tecla apertada
updateShipPosition()	atualiza a posição da nave após o tick do relógio
insideKeeper()	mantém a nave dentro das dimensões do cenário
shootFromShip()	responsável por fazer a nave atirar
gotDamagedShip()	diminui a vida da nave quando ela é atingida
isShipAlive()	verifica se a vida da nave é maior que zero
updateScore()	atualiza o score da nave
updateFocus()	atualiza o valor da barra de slow-motion
renderShip()	desenha a nave
getAimV()	calcula vetor da mira
renderAim()	desenha a mira
renderShipFinal()	desenha a nave rotacionada e redimensionada

Enemy

Arquivos: “enemy.h”, “enemy.c”.

Atributos	
life	um inteiro de 0 a 100 que diz o quanto a nave tem de “vida”
position	do tipo Position (vide utils), controla a posição da nave no espaço
radius	raio do inimigo

Funções	
createEnemy()	cria o inimigo
killEnemy()	libera o espaço da memória utilizado pelo inimigo
shootFromEnemy()	responsável por fazer o inimigo atirar
shouldShoot()	diz se a nave deve atirar ou não
gotShotEnemy()	diminui a vida do inimigo quando ele é atingido
isEnemyAlive()	verifica se a vida do inimigo é maior que zero
renderEnemy()	desenha o inimigo

Shot

Arquivos: “shot.h”, “shot.c”.

Atributos	
position	do tipo Position, diz a posição do projétil no espaço
velocity	do tipo Velocity, diz a orientação e velocidade do projétil
damage	diz a potência do projétil

Funções	
createShot()	cria um novo projétil
freeShot()	libera o espaço alocado por um projétil
updateShot()	atualiza a posição do projétil de acordo com o tick do relógio
renderShot()	desenha o tiro

Hud

Arquivos: "hud.h", "hud.c"

Funções	
renderHud()	desenha a hud
renderSpeed()	desenha o velocímetro
renderLife()	desenha a barra de vida
renderFocus()	desenha a barra de slow-motion
renderScore()	desenha o score

EnemyQueue

Arquivos: "enemyQueue.h", "enemyQueue.c".

Atributos	
*first	ponteiro para o primeiro inimigo da queue
*last	ponteiro para o último inimigo da queue
*head	ponteiro para o nó da cabeça da queue
*lastNode	ponteiro para o último nó da queue

Funções	
createEnemyQueue()	cria a queue
removeEnemyNode()	remove o inimigo passado como argumento
enqueueEnemy()	coloca o inimigo na queue
dequeueEnemy()	tira o inimigo da queue
isEnemyQueueEmpty()	verifica se a queue está vazia
renderEnemyQ()	desenha todos os elementos da fila

ShotQueue

Arquivos: "shotQueue.h", "shotQueue.c".

Atributos	
*first	ponteiro para o primeiro inimigo da queue
*last	ponteiro para o último inimigo da queue
*head	ponteiro para o nó da cabeça da queue
*lastNode	ponteiro para o último nó da queue

Funções	
createShotQueue()	cria a queue
removeShotNode()	remove o tiro passado como argumento
enqueueShot()	coloca o tiro na queue
dequeueShot()	tira o tiro da queue
isShotQueueEmpty()	verifica se a queue está vazia
freeShotQueue()	libera espaço ocupado pela fila
updateShotQueue()	atualiza fila
renderShotQ()	desenha todos os elementos da fila

Configuration

Arquivos: "configuration."

Utils	
WIN_HEIGHT	altura da janela
WIN_WIDTH	largura da janela
HALF_WIN_HEIGHT	metade da altura da janela
HALF_WIN_WIDTH	metade da largura da janela
BUFFER_SIZE	tamanho do buffer do cenário
MAX_DISTANCE_BETWEEN_ENEMIES	Distancia maxima entre dois inimigos
MAX_SHIP_SHOTS	Número máximo de tiros que a nave pode atirar
GRAVITY	valor da gravidade
ENEMY_SHOCK_POWER	valor retirado da vida em colisões da nave

Ship	
INITIAL_VELOCITY	velocidade inicial da nave
MOVING_FACTOR	o quanto muda a direção da nave quando ela vira
VELOCITY_FACTOR	o quanto muda a velocidade da nave quando ela freia ou acelera
MAX_VELOCITY	limita a velocidade da nave
SHIP_SHOT_NORM	velocidade do tiro que sai da nave

Enemy	
SHOOTABLE_DISTANCE	distância mínima até a nave para a qual o inimigo atira
ENEMY_SHOT_VELOCITY	velocidade do tiro que sai do inimigo