River Raid - 1ª fase

Abril 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).

Descrição dos arquivos e funções

Ship

Os arquivos ship.c e ship.h são responsáveis por controlar a nave.

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

Funções	
shootFromShip()	responsável por fazer a nave atirar
createShip()	cria a nave
moveShipHorizontally()	move a nave no eixo X
moveShipVertically()	move a nave no eixo Y
changeShipSpeed()	muda a velocidade da nave
updateShipPosition()	atualiza a posição da nave após o tick do relógio
killShip()	libera o espaço da memória usado pela nave
gotShotShip()	diminui a vida da nave quando ela é atingida
isShipAlive()	verifica se a vida da nave é maior que zero

Constantes	
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_XY_ORIENTATION	limita o quanto a nave pode virar
MAX_VELOCITY	limita a velocidade da nave

Enemy

Os arquivos enemy.h e enemy.c são responsáveis por controlar os inimigos.

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
precision	um inteiro de 0 a 10 que controla a eficácia dos tiros do inimigo

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

Constantes	
SHOOTABLE_DISTANCE	distância mínima até a nave para a qual o inimigo atira

Shot

Os arquivos shot.h e shot.c são responsáveis por controlar os projéteis.

Atributos	
shotPosition	do tipo Potision, diz a posição do projétil no espaço
shotVelocity	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
computeShotNorm()	atualiza a variável que guarda a norma do vetor velocidade do protétil

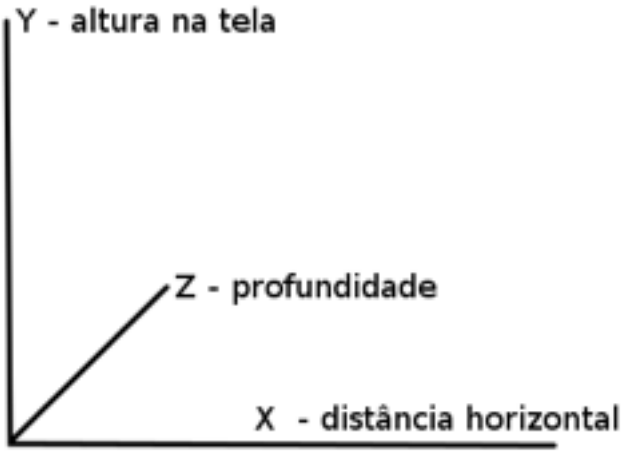
Constantes	
SHOOT_NORM	indica a norma do vetor velocidade do projétil

Observações:
Não definimos se o projétil vai ter mudança de trajetória, então não sabemos se vamos usar a definição SHOOT_NORM ou se vamos usar a variável shotNorm. Uma mudança de trajetória do projetil seria interessante para criar um projétil que vai de encontro com a nave ou inimigo.

Utils

Os arquivos utils.h e utils.c reunem funções e variáveis relevantes a mais de um arquivo do projeto.

O arquivo 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	
distance()	calcula a distancia entre duas Position
isItAtScreen()	verifica se uma Position está na frente da nave
spaceTimeEquation()	atualiza uma Position de acordo com sua Velocity

Variáveis Globais	
clockTick	valor em segundos do tick do relógio
ySize	altura da janela do jogo
xSize	largura da janela do jogo

Cenario

Os arquivos cenario.c e cenario.h reúnem informações sobre a organização do cenário do jogo. São também responsáveis por gerenciar a fila de inimigos.

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
verifyShotColision()	verifica se o tiro colidiu com algum inimigo

Programa parcial

Composto pelo arquivo main.c, é a parte que junta todas as funções e arquivos descritos anteriormente. Para isso há a implementação de algumas funções que nos permite visualizar as estruturas criadas. Também há o uso de um enum para simular a leitura de teclas que deveria ser feito.

Com essa parte do programa, simulamos a criação de um cenário e sua fila de inimigos, criação de uma nave e um tiro, e a atualização dessas estruturas num loop infinito. Utilizando os números 0, 1, 2, 3, 4, 5, controlamos as ações de setas direcionais, espaço e click do mouse, assim como também lemos do teclado a posição do mouse para ser usado na mira do tiro.

Até o momento é possível nos movimentarmos dentro dos limites do cenário, perder vida caso colidamos com os inimigos e podemos atirar nos inimigos até que eles sejam destruídos.

Principais dificuldades

As principais dificuldades que encontramos no desenvolvimento do projeto foram:

- Falta de tempo devido a semana de provas.
- Decidir se os projéteis mudam de trajetória ou não.
- Decidir a orientação inicial de um projétil.