R-type

Este arquivo tem como objetivo explicar de forma simples a lógica e mostrar a estrutura do código do jogo.

Estrutura

Este jogo foi programado na linguagem C e com auxílio da biblioteca Allegro, que é muito utilizada no desenvolvimento de diversos jogos.

Para melhor compreensão do código serão apresentadas as estruturas, funções e as structs que foram necessárias para criar o jogo. Além disso, também citarei as variáveis e constantes. Dessa forma, é válido destacar que separei as funções em bibliotecas para entender de forma mais clara meu código.

Bibliotecas

Nesse tópico irei citar as bibliotecas que criei para colocar as funções e assim facilitar as alterações e, por conseguinte, melhorar a organização.

configuaracoes.h

Essa parte é responsável pela declaração das variáveis globais que são utilizadas no jogo. Linha 3 até 28: são declaradas essas variáveis. Dentre elas estão as dimensões da nave, velocidade, imagem e condição para colisão. Sob essa perspectiva, também é implementada a quantidade de inimigos que o jogo vai gerar nas funções posteriores.

arquivo

rand.h

A função rand que implementei nesta biblioteca foi inspirada na aula do Olmo que aborda como implementar uma função que gera números aleatórios. Dessa forma, ela foi adaptada para gerar os inimigos.

nave.h

Da linha 1 até 8: é definida uma struct nave e é passado suas dimensões, cor e velocidade. Além disso, partindo da linha 10 a 61 são criadas funções que possuem a intensão de desenhar a nave na tela e limitar seus movimentos no limite no monitor.

void **initNave**(Nave \***nave**): Essa função possui a finalidade de definir as dimensões que a nave vai possuir assim como, sua velocidade e cor.

void **desenhaNave**(Nave **nave**): Por meio da **al\_draw\_bitmap** é passada a o nome da foto que vou utilizar na imagem e suas coordenadas.

A terceira e quarta função servem para atualizar e limitar a nave respectivamente.

inimigo.h

Na struct é passada algumas coisas importantes para criar o inimigo nas funções seguintes, como dimensões x, y e também velocidade e cor.

void **initInimigo**(Inimigo **inimigo[]**, int **tamanho**):essa função passa a struct inimigo e seu tamanho limite. Diante disso, é passada nessa função a velocidade do inimigo o seu tamanho que é formado de forma aleatória pela função rand. Além disso, é definida o tamanho de suas bordas, uma verificação se o inimigo está ativo e por fim, sua cor que é atribuída de maneira aleatória.

void liberaInimigo: possui a finalidade de liberar de forma aleatória os inimigos de diferentes partes da tela do jogo**.**

void atualizarInimigo: cria um loop para atualizar o inimigo e verifica se o inimigo ultrapassou o limite de tela.

void desenhaInimigo: Por fim, essa função através da al\_draw\_filled\_circle desenha o círculo que representa o inimigo.

tiro.h

Struct tiro: passa suas dimensões, velocidade, cor e um booleano que verifica se o tiro está ativo.

void **initTiro**(Tiro \***tiro**, Nave \***nave**): Define o lugar que o tiro vai ficar, sua velocidade, cor e imputa falso para a variável ativo.

void **desenhaTiro**(Tiro **tiro**): através também da al\_draw\_filled\_circle desenha o tiro.

void **atualizaTiro**(Tiro \***tiro**, Nave \***nave**): atualiza a posição do tiro quando sua velocidade for diferente de 5.

void **saiTiro**(Tiro \***tiro**, Nave \***nave**, Bloco \***bloco**): e por último essa função libera o tiro quando sua velocidade for igual a 5.

cenario.h

void intGlobais(): atribui de forma aleatória as cores que vão compor o fundo do cenário.

void desenhaCenario(): Essa função pega COR\_CENARIO definida na função anterior e usa al\_clear\_to\_color para definir a cor do fundo do jogo.

pontuação.h

void **adicionapontos**(): Nessa função eu somo dez pontos cada vez que tiro colide com o inimigo.

void **pontuacao**(ALLEGRO\_FONT \***size\_32**): Neste caso essa função desenha na tela a pontuação momentânea do jogador.

void **recorde**(): Para finalizar, essa última função eu verifico se ouve a colisão da nave com um inimigo ou bloco. Dessa feita, eu desenho no fim do jogo por 3 segundos a sua pontuação no jogo e seu recorde no jogo em geral e se sua pontuação for maior que o recorde registrado, ira aparecer na tela “Novo Recorde” e sua respectiva pontuação.

colisões.h

void **colisoesNave**(Inimigo **inimigo[]**, Bloco **bloco**, Nave **nave**):Verifica se as coordenadas nave colidem com as do inimigo ou com os blocos e encerra o jogo quando for verdadeiro.

void **TiroColideInimigo**(Tiro **tiro**, Inimigo **inimigo[]**, int **tamanho**, Nave **nave**): essa função é responsável por detectar a colisão entre o tiro e o inimigo e adiciona pontos quando isso acontecer.

void **colideInimigoBloco**(Inimigo **inimigo[]**, Bloco **bloco**): Elimina os inimigos da tela quando eles colidem com o bloco.

Inicializações.h

Essa parte é possui a finalidade iniciar as variáveis inteiras globais e é passada as funções necessárias para que o jogo execute.

Includes.h

Nesse arquivo estão os includes da biblioteca alegro 5 e os includes da biblioteca em C. Além disso, todos os arquivos .h são chamados nas suas devidas ordens para o bom funcionamento do jogo.

Arquivo em C

Esse arquivo denominado meurtype é o local na qual está a função int **main**(int **argc**, char \*\***argv**), que é local na qual os outros arquivos saão chamados. Na linha 1 é chamado a “includes.h”, que passa todos arquivos criados e dentro da main é passado os eventos de tempo e também de pressionar as teclas.