Trabalho de comp

Relatório

Nome do trabalho:
CriptoWave
Grupo:
João Henrique Schmidt /119050097
Diego Malta /119044957
Thalles Nonato /119058809
André Uziel /119051475

Proposta

Nossa proposta baseia-se no minigame chamado "Journey of the Prairie King", dentro do jogo Stardew Valley: https://youtu.be/MtdyVHZluAw. O objetivo consiste em derrotar todos os inimigos dentro de uma sala. Os inimigos aparecerão em ondas(grupos), quando o jogador derrotar o último inimigo de uma onda, aparecerá uma nova onda. O jogador ganha ao derrotar o último inimigo da última onda, o jogador perde se for derrotado por algum inimigo. O jogador deverá atirar em um inimigo para derrotá-lo. Se algum inimigo atirar ou tocar no jogador, ele perde um ponto em vida. Se perder toda a vida é fim de jogo.

Bibliotecas

1.**Curses.h**: ncurses consiste em uma biblioteca que permite funcionalidades extras aos terminais e textos.

Variáveis utilizadas:

initscr(); → Inicia a scream padrão do ncurses.
noecho(); →
cbreak(); → Permite que o usuário aperte teclas de saída (ctrl+c)com o objetivo de parar a execução do programa.
mwprint() → Printa as coordenadas do cursor.
wrefresh() → Atualiza a janela.
keypad() → Habilita o uso das setas.
curses_set() → Esconde o cursor da tela padrão.
clear() → Limpa a janela após a atualização da janela (função refresh).
getch() → Espera até o usuário apertar a tecla, e após isso avança.

2. **Time.h**: a biblioteca time.h permite ao desenvolvedor manipular datas e horários.
clock_gettime() → No primeiro parâmetro, ele emite a posição arbitrária do tempo única e imutável do computador. E no segundo parâmetro, ele terá o valor do tempo.

3. **Unistd.h:** Unistd.h é uma biblioteca que permite manipulação de diversas funções. O cabeçalho com suas alterações aparece ao final.
sleep(); → suspende a execução após um intervalo de tempo.

4. **Stdlib.h:** A biblioteca stdlib.h serve para emular do prompt do sistema operacional ao qual estamos programando.

malloc(); → a função malloc aloca espaços para um bloco de bytes consecutivos na memória RAM.

5. **Stdio.h:** A biblioteca <stdio.h> cuida da parte de entrada e saída de dados do programa.

6. **Ctype.h**: Permite a utilização da função isalnum(int), retorna true se o valor do parâmetro é uma letra ou número.

Funções do jogo

Enemy.h:

1) void controle_inimigo0:
A cada segundo é atualizado o movimento do inimigo “0”: Calcula a coordanada do jogador para se aproximar dele.

2) void controle_inimigo_1:
Iguala o eixo horizontal para poder se aproximar do jogador.

3) void controle_inimigo_2:
Iguala o eixo vertical para após isso se aproximar do jogador pelo eixo horizontal

4) void colisao_tiro_inimigo_0e1:
Calcula se a coordenada dos dispáros é igual a do inimigo.

5) void colisao_tiro_inimigo_2:
Mesma função que “colisao_tiro_inimigo_0e1” excetuando o fato da posição em que o inimigo é deletado.

6) void colisao_player_inimigos_0e1:
Quando o inimigo fica na mesma coordenada que o jogador.

7) void colisao_player_inimigos2:
Mesmas funcionalidades de “colisao_player_inimigo_0e1”.

Jogo.h:

1) void execute_wave:
Função ativa as ondas de tropas de inimigas.

2) void write_your_initials:
Função para escrever o nome do jogador.

Player.h:

1)void adciona_tiro_ao_jogador:
Quando a função é chamada, um vetor tiro que esta morto e vai torna-lo vivo, e colocar na posição do jogador.

2) void clearPlayerTrack:
Limpa o trajeto do jogador.

3) void movUp:
Trajeto no eixo y superior.

4) void movDown:
Trajeto no eixo y inferior.

5) void movLeft:
Trajeto no eixo x negativo.

6) void movRight:
Trajeto no eixo x positivo.

7) void desenhaplayer:
Printa o trajeto do jogador.

8) void desenhatiroY:
Printa o trajeto do tiro no eixo y.

9) void desenhatiroX:
Printa o trajeto do tiro no eixo x.

10)void colisao_tiro:
Colide o tiro com a parede.

Time_Elapsed.h:

1) **float time_elapsed:**

Essa função retorna falso se o tempo (primeira variável) passado desde a última vez que retornou true é menor que o segundo parâmetro. Caso seja maior, retorna true e o valor da primeira variável é resetado

Pseudocódigo

Algoritmo CriptoWave

→ Início de programa

→ ncurses iniciado

→ caracteres são desabilitados

→ menu com opções para o usuário aparece na tela

→ tela do jogo é inicializada

→ tropa é inicializada e se movimenta em direção ao jogador

→ novos monstros aparecem se a wave acabar

→ pontuação computa o número de abates do jogador

→ se a posição do monstro = posição do jogador

-jogador perde 1 vida

→ se a vida do jogador chegar a 0

- jogo acaba

-tela de fim de jogo aparece

→ se posição de tiro do jogador = posição do monstro

- monstro morre

→ caso contrário novas tropas aparecem até o jogador ficar sem vida

→ Se o processo for repetido 5 vezes, jogador vence.

→ Output score do jogador = 3*inimigos destruídos - 10*vidas perdidas

Testes realizados e resultados obtidos

O trabalho passou por várias etapas em sua confecção, na fase alpha do projeto testamos diversas formas de tornar o jogo acessível para duas pessoas. Infelizmente a tecnologia que o terminal nos fornece em consonância com pouco tempo de estudo fez com que não fosse possível colocar os dois players ,visto que não possuíamos tempo para estudar a chamada de dois inputs ao mesmo tempo e com isso nos trouxe resultados que não eram os esperados. Avançando no desenvolvimento do jogo o grupo se deparava com testes que mostravam outros bugs no código. Além disso, também houve problemas no que tangia ao disparo do jogador e à movimentação das tropas, pois o terminal não gerava as imagens suficientemente rápidas de forma a dar a impressão de movimento. Após pesquisas chegamos a resolução desses problemas. Novas soluções foram projetadas e chegamos a resultados satisfatórios na opinião do grupo.

Descrição dos casos testes projetados

Tentativa de multiplayer:

Na fase alpha do desenvolvimento do jogo, chegamos a desenvolver dois jogadores. Nos testes percebemos problemas em relação as simetrias dos eixos, pois o jogador que controla o player 1 ao invés de sobrepor-se ao player 2, ele acabava por empurrar o player 2, isso ocorria devido ao processamento do player 1 ser mais rápido que o do player 2. Após isso retiramos o player 2 por não ter tempo para estudar e resolver esse bug.

Problemas com tiros:

O grupo teve dificuldades com o número de tiros disparados pelo jogador. Isso porque, passado um certo tempo, após ultrapassar o limite de tiros o programa travava com o erro de segmentation (core dumped). O bug era causado pela função desenhartiro, que n existe mais e a função colisao_inimigo, que também n existe mais. As duas funções tentavam ler tiros cuja posição no vetor tiros_jogador era maior que o tamanho dele. Acessando memória ilegal e, assim, dando segmentation fault.

Menu:

Na confecção do menu, ao testarmos suas funcionalidades reparamos que a tecla “Enter” não estava funcionando, mesmo utilizando de forma correta as opções de “Keypad” da biblioteca do ncurses. Como solução, modificamos as linhas de código referente ao Keypad do ncurses por cases, onde conseguimos trocar o “Enter” por “\n” e obtivemos o mesmo resultado das funções keypad.

Dificuldades encontradas e conclusão:

No geral, o grupo sentiu uma dificuldade inicial por não estar acostumado a utilizar o ncurses, foi a primeira experiência com a biblioteca para todos os integrantes do grupo. Além disso, apesar dessa biblioteca ser conhecida por desenvolvedores Linux, o conteúdo especificamente para a linguagem C é escasso . Entretanto, com uma pesquisa mais elaborada (principalmente em busca por conteúdos em inglês) e ajuda em grupos de computação da própria universidade nós conseguimos trabalhar de forma satisfatória com a biblioteca. Como citado acima, houve bugs de resolução simples que conseguimos resolver em pouco tempo como também houveram bugs mais elaborados e que nos exigiram soluções mais criativas para resolver-los.

Testes e progresso diários:

André 11-13/6/19
Gênesis 1:3
Disse André: "Haja luz", e houve luz.
criada toda a estrutura básica do ncurses (até então toda na mesma janela), além de cores, controles do jogador, inimigos, tiros, colisões e estruturas
João 14/6/19:
removi altura e largura, cada janela terá altura e largura especificas. Jogadores agora são inicializados como ponteiros player1 e player2. Movimentos e limites de player1 e player2 são a mesma função, assim o pc n precisa calculcar as condições de limite sempre.

Jogador e inimigos agora são inicializados como ponteiros. Muitos "if"s foram alterados para swtich. Colocada janela de batalha "playerWin"
João 15/6/19:
Criei a função time_elapsed(float*,float), retorna verdadeiro se o tempo (segundo parâmetro) for ultrapassado e zera o valor da primeira variavel. Se for falso retorna 0.
wclean() removida, entidades que se movimentam apagarão seus rastros.
João 19/6/19:
typedef struct _tiro criada, agora os tiros são uma struct que contêm a localização e a "existência" do próprio tiro. Os tiros estão em um vetor de ponteiros, cada tiro é modificado como um ponteiro.

Diego e thalles 19/6/19:
Menu e créditos adicionados, utilizado a função sprintf para armazenar as opções de menu em list[]. Função main usada pra rodar o jogo foi nomada como jogo() e main()
será o menu.

João 20/6/19:
Bug do número de tiros, depois de um certo tempo, após ultrapassar o limite de tiros o programa trava com erro segmentation fault (core dumped). Colisao monstro e tiro refeita. Removido todas as funçoes sprintf, removido qtiroesquerda, qtirodireita, qtiroacima, qtiroabaixo, tiroaesquerdax, tiroadireitay, etc, do jogador.
Adicionado pontuação no arquivo score.txt, adicionado créditos, adicionado limite de tela para tirs. Consterado bug de traços no canto esquerdo da tela.

João 21/6/19:
Bug de tiros consertado. Adicionado inimigo0, 1 e 2. 1 é o inimigo anterior, 2 é uma variação dele que se move no eixo y. Inimigo0 é totalmente novo e calcula a distância exata do jogador. Adicionado ondas, adiciondo fim de jogo, adicionado tbm colisao entre o inimigo com o tiro e o player, adionado vida, adicionado score, adicionado janela de status do jogador, adicionado organizador de pontuação. Definidos quantidade máxima de cada um dos inimigos. Player 2 REMOVIDO

Diego e Thalles 25/06/19:
Meu Deus, que aventura! Passamo mal programando as waves. Toda hora cometiamos erros pequenos, como esquecer de mudar valores de variáveis e ";". Fizemos a tela de vitória, estilizamos a tela de highscore e resolvemos bugs dela. Tiramos o bug da tela inicial de parar de mostrar o título depois da primeira vez que o jogo é iniciado. Fizemos a terceira, quarta e quinta waves. Relatório feito