DRAKE & JOSH - SUSHI APOCALYPSE

Aplicação do conceito de filas em um jogo

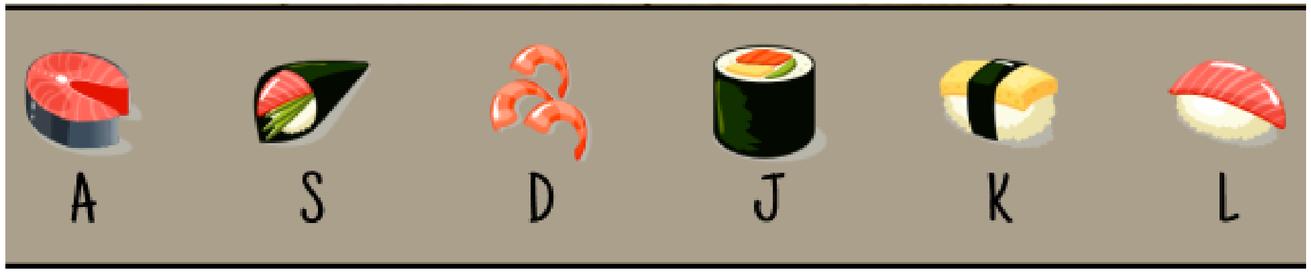
Gustavo Fernandes
Johann Lambert
Victor Hugo Ferreira
Wendel Petta

Descrição

- Inspirado no episódio "Eu Amo Sushi" do seriado "Drake & Josh"
- Objetivo: coletar os sushis dentro do tempo limite estabelecido;
- Estratégia: agilmente comande os botões correspondentes aos elementos, gerados aleatoriamente em uma fila

Mêcanicas de jogo

- Identificar as teclas de atalhos e coletar a maior quantidade de sushis dentro do tempo limite
- Caso colete todos os elementos gerados na fila, o jogador avança um nível, aumentando a dificuldade do jogo
- Popup’s aparecerão aleatoriamente na tela, te garantindo mais tempo e pontos
- Erros nos inputs resultam em uma penalidade no tempo restante

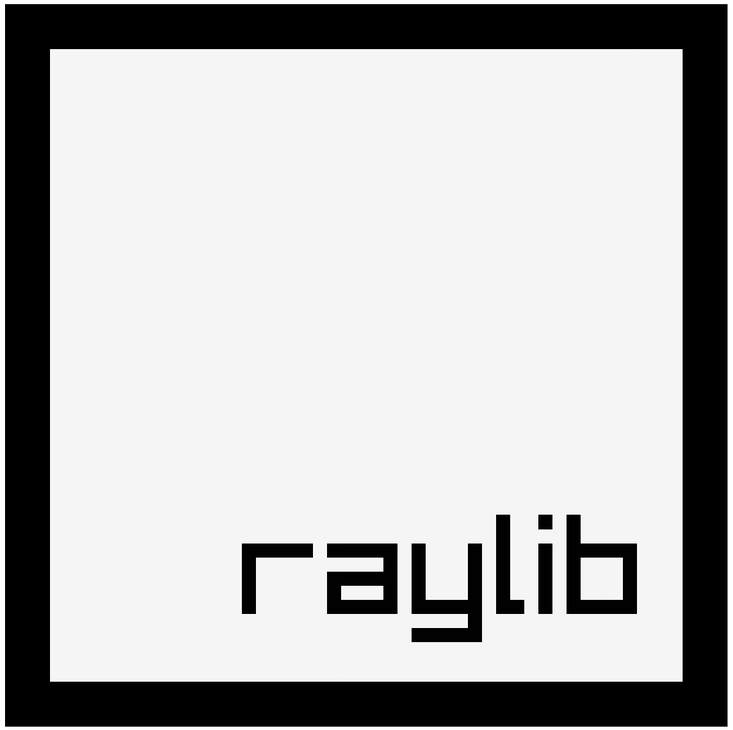


Mêcanicas de jogo

- Ao final do pedido, se o jogador conclui-lo, iniciará uma contagem regressiva antes de prosseguir para o próximo pedido
- Caso não conclua a tempo, é decretado fim de jogo

Biblioteca gráfica

- O jogo foi desenvolvido utilizando a biblioteca raylib
- A biblioteca utiliza a linguagem C, fornecendo uma janela que permite o uso de elementos gráficos 2D e 3D
- Para o jogo os principais módulos utilizados foram os “core” e “textures”



Principais Funções

- enfileirar
- desenfileirar
- geraItens
- checkKeyPressed
- comparaInput
- printFila
- updateSushiPositions

enfileirar

- Semelhante à função vista em aula
- Difere na quantidade de dados contidos no Nó

```c
void enfileirar(No **fila, char item, int posDesloc){
    No *aux, *novo = malloc(sizeof(No));
    if(novo){
        novo->item = item; // "ID" do item
        novo->posX = 1220 + posDesloc; //coordenada x de posição na tela
        novo->posY = 505; //coordenada y de posição na tela
        novo->proximo = NULL;
        if(*fila == NULL)
            *fila = novo;
        else{
            aux = *fila;
            while (aux->proximo)
                aux = aux->proximo;
            aux->proximo = novo;
        }
    }
}
```

desenfileirar

- Identico ao que foi visto em aula

```
No* desenfileirar(No **fila){
    No *remover= NULL;

    if(*fila){
        remover = *fila;
        *fila = remover->proximo;
    }
    return remover;
}
```

geraItens

- Função responsável por gerar aleatoriamente itens contidos na fila através da função enfileirar
- É chamada no começo do jogo e a cada vez que o jogador conclui uma das filas

```
void geraItens(No** fila, int dificuldade){ // função que gera os items com
    for(int i = 0; i<dificuldade; i++){
        switch(rand()%6){
            case 0:
                enfileirar(fila, 'A', i*espacamentoInicialSushis);
                break;
            case 1:
                enfileirar(fila, 'S', i*espacamentoInicialSushis);
                break;
            case 2:
                enfileirar(fila, 'D', i*espacamentoInicialSushis);
                break;
            case 3:
                enfileirar(fila, 'J', i*espacamentoInicialSushis);
                break;
            case 4:
                enfileirar(fila, 'K', i*espacamentoInicialSushis);
                break;
            case 5:
                enfileirar(fila, 'L', i*espacamentoInicialSushis);
                break;
        }
    }
}
```

checkKeyPressed

- Gerente de inputs do jogo
- Recebe um ponteiro para a fila para envia-lo como parâmetro para a próxima função

```
void checkKeyPressed(No** fila, int *pontuacao, Timer* playingTimer, Event* eventoPopup){ // função
    int input = GetKeyPressed();

    if(eventoPopup->isActive && input == KEY_E && comparaInput(fila, 'E', eventoPopup)){
        playingTimer->Lifetime += 3;
        eventoPopup->isActive = false;
        *pontuacao += 3 * pontosPorAcerto;
    }
    else if(eventoPopup->isActive && input == KEY_U && comparaInput(fila, 'U', eventoPopup)) {
        playingTimer->Lifetime += 3;
        eventoPopup->isActive = false;
        *pontuacao += 3 * pontosPorAcerto;
    }
    else if(input == KEY_A && comparaInput(fila, 'A', eventoPopup)) *pontuacao += pontosPorAcerto;
    else if(input == KEY_S && comparaInput(fila, 'S', eventoPopup)) *pontuacao += pontosPorAcerto;
    else if(input == KEY_D && comparaInput(fila, 'D', eventoPopup)) *pontuacao += pontosPorAcerto;
    else if(input == KEY_J && comparaInput(fila, 'J', eventoPopup)) *pontuacao += pontosPorAcerto;
    else if(input == KEY_K && comparaInput(fila, 'K', eventoPopup)) *pontuacao += pontosPorAcerto;
    else if(input == KEY_L && comparaInput(fila, 'L', eventoPopup)) *pontuacao += pontosPorAcerto;

    else if(input != 0) playingTimer->Lifetime -= penalidadePorErro;
}
```

comparaInput

- Relaciona o input da função anterior e valida com os elementos da fila e do jogo
- Utiliza da função desenfileirar após validar o input em relação ao primeiro elemento da fila

```
bool comparaInput(No** fila, char input, Event* eventoPopup){

    if(eventoPopup->isActive && eventoPopup->eventType == input) return true;

    // verifica se input equivale a primeiro elemento da fila
    if(*fila){
        No *remover, *aux = *fila;
        if(input == aux->item){
            remover = desenfileirar(fila);
            free(remover);
            return true;
        }
    }
    return false;
}
```

printFila

- Similar à função de print desenvolvida em aula
- Difere no ponto em que no lugar de um printf, utiliza uma outra função responsável por renderizar assets na tela

```
void printFila(No** fila, Texture2D* sushis){ // p
    No* aux = *fila;
    if(*fila){
        drawSushi(aux, 18, sushis);
        while(aux->proximo){
            drawSushi(aux->proximo, 18, sushis);
            aux = aux->proximo;
        }
    }
}
```

updateSushi Positions

- Itera pela fila de forma similar à função de print
- Difere em atualizar os dados de posição X e Y dos nós, com base em uma velocidade em que devem se mover pela tela e limites de tela e de outros sushis

```
void updateSushiPositions(No** fila, int velocidade){ //itera pela fila e atualiza sua
    No* aux = *fila;
    if(*fila){
        if(aux->posX >= limiteEsteira) aux->posX -= velocidadeEsteira + velocidade/4;
        while(aux->proximo){
            if(aux->proximo->posX >= (aux->posX + espacamentoMinimoSushis)){
                aux->proximo->posX -= velocidadeEsteira + velocidade/4;
            }
            aux = aux->proximo;
        }
    }
}
```

MUITO OBRIGADO!

Drake & Josh
Sushi Apocalypse