|  |
| --- |
| CSE2003 – Data structures & algorithms |
| SNAKE GAME USING QUEUE |
| PROJECT REPORT |
|  |
| **Nithesh Gurudaas – 18BCE1201**  **+91 99947 09300**  [kg.nitheshgurudaas2018@vitstudent.ac.in](mailto:kg.nitheshgurudaas2018@vitstudent.ac.in) |
|  |
| **Aditya Ganti – 18BCE1292**  **+91 8390785130**  aditya.ganti2018@vitstudent.ac.in |
| **10/11/2019** |

|  |
| --- |
| Dr. Oswald |

## MOTIVATION:

The motivation behind this project was to learn game developing. We are aspiring game developers and we want to have a good, strong base for our future.

## PROBLEM STATEMENT:

To construct a snake game in Unity engine using queue data structure. The game has to have its normal functions such as creating the level grid, spawning food, increasing body size of snake after food is eaten, increasing speed of game, keeping track of score, checking if snake has died or not. All of these functions have to be integrated to get a fully functional game.

## ALGORITHM:

1. Create the level grid and randomly spawn food anywhere on the grid. Create the snake and spawn it at the origin. Set the snake\_body\_size to 0, and score to 0.
2. Make the snake move automatically in the direction that is facing. Check if the snake’s position and food position is the same. If yes, then increase the snake body size and append the position to the snake move position vector list(Included in snake) and increase the score of the snake.
3. Check if the user has input any direction keys. We will have certain constraints for the key pressed, such as: if the snake is facing left and the user enters right, then the snake cannot turn 180, so we’ll keep a few similar restrictions.
4. Check if the snake head position and any body part’s position is the same or not. If yes, then end the game and show game over scene. Else go to step 2.

The snake will be a queue of size snake\_body\_size, and it will contain the locations of the last snake\_body\_size number of places the snake head visited. We will display snake body sprites in all of those positions to indicate that the snake is moving.

## DATA STRUCTURE USED/STRATEGY:

We have used a queue data structure here to keep track of the positions that the snake has visited. We will have a queue of size same as the snake body size. The queue will be updated whenever the snake moves. We will append the latest position to the front of the queue and remove the elements after position of snake body size (i.e., elements after arr[snake\_body\_size]). We will keep on doing this and check if any body part has the position same as the head, and if head position and body part position is same, we’ll end the game.

## FUTURE WORK:

We hope to develop newer, innovative games in the future. We developed this game so that we can get to know the basics of Unity Engine and basics of C# programming. We hope to create more games in the coming days.

# CODE

**GAME ASSETS:**using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class GameAssets : MonoBehaviour

{

public static GameAssets i;

private void Awake() {

i = this;

}

public Sprite snakeHeadSprite;

public Sprite foodSprite;

public Sprite snakeBodySprite;

}

**GAME HANDLER:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using CodeMonkey;

using CodeMonkey.Utils;

public class GameHandler : MonoBehaviour

{

private static GameHandler instance;

private static int score;

[SerializeField] private Snake snake;

private LevelGrid levelGrid;

private void Awake()

{

instance = this;

InitializeStatic();

}

void Start()

{

Debug.Log("GameHandler.Start");

levelGrid = new LevelGrid(20, 20);

snake.Setup(levelGrid);

levelGrid.Setup(snake);

}

private static void InitializeStatic()

{

score = 0;

}

public static int GetScore()

{

return score;

}

public static void AddScore()

{

score += 100;

}

public static void SnakeDied()

{

GameOverWindow.ShowStatic();

}

}

**GAME OVER WINDOW:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using CodeMonkey.Utils;

public class GameOverWindow : MonoBehaviour

{

private static GameOverWindow instance;

private void Awake()

{

instance = this;

transform.Find("RetryBtn").GetComponent<Button\_UI>().ClickFunc=()=>

{

Loader.Load(Loader.Scene.GameScene);

};

Hide();

}

private void Show()

{

gameObject.SetActive(true);

}

private void Hide()

{

gameObject.SetActive(false);

}

public static void ShowStatic()

{

instance.Show();

}

}

**LEVEL GRID:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using CodeMonkey;

public class LevelGrid

{

private Vector2Int foodGridPosition;

private GameObject foodGameObject;

private int width;

private int height;

private Snake snake;

public LevelGrid(int width,int height)

{

this.width = width;

this.height = height;

}

public void Setup(Snake snake)

{

this.snake = snake;

SpawnFood();

}

private void SpawnFood()

{

do

{

foodGridPosition = new Vector2Int(Random.Range(0, width), Random.Range(0, height));

} while (snake.GetFullSnakePositionList().IndexOf(foodGridPosition)!=-1);

foodGameObject = new GameObject("Food", typeof(SpriteRenderer));

foodGameObject.GetComponent<SpriteRenderer>().sprite = GameAssets.i.foodSprite;

foodGameObject.transform.position = new Vector3(foodGridPosition.x, foodGridPosition.y);

}

public bool TrySnakeEatFood(Vector2Int snakeGridPosition)

{

if (snakeGridPosition == foodGridPosition)

{

Object.Destroy(foodGameObject);

SpawnFood();

GameHandler.AddScore();

return true;

}

else

return false;

}

public Vector2Int ValidateGridPosition(Vector2Int gridPosition)

{

if(gridPosition.x<0)

{

gridPosition.x = width - 1;

}

if(gridPosition.y<0)

{

gridPosition.y = height - 1;

}

if(gridPosition.x>=width)

{

gridPosition.x = 0;

}

if(gridPosition.y>=height)

{

gridPosition.y = 0;

}

return gridPosition;

}

}

**LOADER:**

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public static class Loader

{

public enum Scene{

GameScene,

Loading

}

private static Action loaderCallbackAction;

public static void Load(Scene scene)

{

loaderCallbackAction = () =>

{

SceneManager.LoadScene(scene.ToString());

};

SceneManager.LoadScene(Scene.Loading.ToString());

}

public static void LoaderCallback()

{

if(loaderCallbackAction!=null)

{

loaderCallbackAction();

loaderCallbackAction = null;

}

}

}

**LOADER CALL BACK:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class LoaderCallback : MonoBehaviour

{

private bool firstUpdate = true;

private void Update()

{

if(firstUpdate)

{

firstUpdate = false;

Loader.LoaderCallback();

}

}

}

**SNAKE:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using CodeMonkey;

using CodeMonkey.Utils;

public class Snake : MonoBehaviour

{

private enum Direction

{

Left,

Right,

Up,

Down

}

private enum State {

Alive,

Dead

}

private State state;

private Direction GridMoveDirection;

private Vector2Int GridPosition;

private float GridMoveTimer;

private float GridMoveTimerMax;

private LevelGrid levelGrid;

private int snakeBodySize;

private List<SnakeMovePosition> snakeMovePositionList;

private List<SnakeBodyPart> SnakeBodyPartList;

public void Setup(LevelGrid levelGrid)

{

this.levelGrid = levelGrid;

}

private void Awake()

{

GridPosition = new Vector2Int(10, 10);

GridMoveTimerMax = .125f;

GridMoveTimer = 0f;

GridMoveDirection = Direction.Right;

snakeBodySize = 0;

snakeMovePositionList = new List<SnakeMovePosition>();

SnakeBodyPartList = new List<SnakeBodyPart>();

state = State.Alive;

}

private void Update()

{

switch (state)

{

case State.Alive:

HandleInput();

HandleGridMovement();

break;

case State.Dead:

break;

}

}

private void HandleInput()

{

if (Input.GetKeyDown(KeyCode.UpArrow))

{

if (GridMoveDirection != Direction.Down)

{

GridMoveDirection = Direction.Up;

}

}

if (Input.GetKeyDown(KeyCode.DownArrow))

{

if (GridMoveDirection != Direction.Up)

{

GridMoveDirection = Direction.Down;

}

}

if (Input.GetKeyDown(KeyCode.RightArrow))

{

if (GridMoveDirection != Direction.Left)

{

GridMoveDirection = Direction.Right;

}

}

if (Input.GetKeyDown(KeyCode.LeftArrow))

{

if (GridMoveDirection != Direction.Right)

{

GridMoveDirection = Direction.Left;

}

}

}

private void HandleGridMovement()

{

GridMoveTimer += Time.deltaTime;

if (GridMoveTimer >= GridMoveTimerMax)

{

GridMoveTimer -= GridMoveTimerMax;

SnakeMovePosition previousSnakeMovePosition = null;

if (snakeMovePositionList.Count > 0)

{

previousSnakeMovePosition = snakeMovePositionList[0];

}

SnakeMovePosition snakeMovePosition = new SnakeMovePosition(previousSnakeMovePosition,GridPosition, GridMoveDirection);

snakeMovePositionList.Insert(0, snakeMovePosition);

Vector2Int gridMoveDirectionVector;

switch (GridMoveDirection)

{

default:

case Direction.Right: gridMoveDirectionVector = new Vector2Int(1, 0); break;

case Direction.Left: gridMoveDirectionVector = new Vector2Int(-1, 0); break;

case Direction.Up: gridMoveDirectionVector = new Vector2Int(0, 1); break;

case Direction.Down: gridMoveDirectionVector = new Vector2Int(0, -1); break;

}

GridPosition += gridMoveDirectionVector;

GridPosition= levelGrid.ValidateGridPosition(GridPosition);

bool snakeAteFood = levelGrid.TrySnakeEatFood(GridPosition);

if (snakeAteFood)

{

snakeBodySize++;

if(GridMoveTimerMax>0.07f)

GridMoveTimerMax -= 0.0025f;

CreateSnakeBody();

}

if (snakeMovePositionList.Count >= snakeBodySize + 1)

{

snakeMovePositionList.RemoveAt(snakeMovePositionList.Count - 1);

}

UpdateSnakeBodyparts();

foreach (SnakeBodyPart snakeBodyPart in SnakeBodyPartList)

{

Vector2Int snakeBodyPartGridPosition = snakeBodyPart.GetGridPosition();

if(GridPosition==snakeBodyPartGridPosition)

{

state = State.Dead;

GameHandler.SnakeDied();

}

}

transform.position = new Vector3(GridPosition.x, GridPosition.y);

transform.eulerAngles = new Vector3(0, 0, GetAnglefromVector(gridMoveDirectionVector) - 90);

}

}

private void CreateSnakeBody()

{

SnakeBodyPartList.Add(new SnakeBodyPart(SnakeBodyPartList.Count));

}

private void UpdateSnakeBodyparts()

{

for (int i = 0; i < SnakeBodyPartList.Count; i++)

SnakeBodyPartList[i].SetSnakeMovePosition(snakeMovePositionList[i]);

}

public Vector2Int GetGridPosition()

{

return GridPosition;

}

public List<Vector2Int> GetFullSnakePositionList()

{

List<Vector2Int> gridPositionList = new List<Vector2Int>() { GridPosition };

foreach (SnakeMovePosition snakeMovePosition in snakeMovePositionList)

{

gridPositionList.Add(snakeMovePosition.GetGridPosition());

}

return gridPositionList;

}

private float GetAnglefromVector(Vector2Int dir)

{

float n = Mathf.Atan2(dir.y, dir.x) \* Mathf.Rad2Deg;

if (n < 0)

n += 360;

return n;

}

private class SnakeBodyPart

{

private Transform transform;

private SnakeMovePosition snakeMovePosition;

public SnakeBodyPart(int bodyIndex)

{

GameObject snakeBodyGameObject = new GameObject("SnakeBody", typeof(SpriteRenderer));

snakeBodyGameObject.GetComponent<SpriteRenderer>().sprite = GameAssets.i.snakeBodySprite;

snakeBodyGameObject.GetComponent<SpriteRenderer>().sortingOrder = -bodyIndex;

transform = snakeBodyGameObject.transform;

}

public void SetSnakeMovePosition(SnakeMovePosition snakeMovePosition)

{

this.snakeMovePosition = snakeMovePosition;

transform.position = new Vector3(snakeMovePosition.GetGridPosition().x, snakeMovePosition.GetGridPosition().y);

float angle;

switch (snakeMovePosition.ReturnDirection())

{

default:

case Direction.Up:

switch(snakeMovePosition.GetPreviousDirection())

{

default:

angle = 0;

break;

case Direction.Left:

angle =0+ 45;

break;

case Direction.Right:

angle = 0-45;

break;

}

break;

case Direction.Down:

switch (snakeMovePosition.GetPreviousDirection())

{

default:

angle = 180;

break;

case Direction.Left:

angle = 135;

break;

case Direction.Right:

angle = 45;

break;

}

break;

case Direction.Left:

switch (snakeMovePosition.GetPreviousDirection())

{

default:

angle = -90;

break;

case Direction.Down:

angle = 135;

break;

case Direction.Up:

angle = 45;

break;

}

break;

case Direction.Right:

switch(snakeMovePosition.GetPreviousDirection())

{

default:

angle = 90;

break;

case Direction.Down:

angle = 45;

break;

case Direction.Up:

angle = -45;

break;

}

break;

}

transform.eulerAngles = new Vector3(0, 0, angle);

}

public Vector2Int GetGridPosition()

{

return snakeMovePosition.GetGridPosition();

}

}

private class SnakeMovePosition

{

private SnakeMovePosition previousSnakeMovePosition;

private Vector2Int gridPosition;

private Direction direction;

public SnakeMovePosition(SnakeMovePosition previousSnakeMovePosition, Vector2Int gridPosition, Direction direction)

{

this.previousSnakeMovePosition = previousSnakeMovePosition;

this.gridPosition = gridPosition;

this.direction = direction;

}

public Vector2Int GetGridPosition()

{

return gridPosition;

}

public Direction ReturnDirection()

{

return direction;

}

public Direction GetPreviousDirection()

{

if (previousSnakeMovePosition == null)

{

return Direction.Right;

}

else

{

return previousSnakeMovePosition.direction;

}

}

}

}

**SCORE WINDOW:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class ScoreWindow : MonoBehaviour

{

private Text ScoreText;

private void Awake()

{

ScoreText = transform.Find("ScoreText").GetComponent<Text>();

}

private void Update()

{

ScoreText.text = GameHandler.GetScore().ToString();

}

}