Report On

2048 Game using Unity

Submitted in partial fulfillment of the requirements of the Course project in Semester VII of Fourth Year Artificial Intelligence and Data Science

by

Divyah Mandavia(Roll No.11)

Khushi Tiwari (Roll No. 28)

Reena Vaidya (Roll No. 31)

Supervisor

Prof. Sejal D'mello

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that the project entitled “2048 Game using Unity” is a bonafide work of "Divyah Mandavia (Roll No.11), Reena Vaidya (Roll No. 31), Khushi Tiwari (Roll No. 28) submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester VII of Second Year Artificial Intelligence and Data Science engineering.

Supervisor
Prof. Sejal D’mello

Dr. Tatwadarshi P. N.
Head of Department

Table of Contents

Problem Statement

The problem statement for creating a 2048 game using Unity for an immersive Virtual Reality (VR) experience involves addressing the challenge of translating the traditional 2D puzzle game mechanics into a 3D VR context. This includes designing an intuitive and comfortable user interface for interaction, implementing effective spatial and gesture controls, optimizing the game for VR performance, and creating an engaging immersive environment. Additionally, the problem statement encompasses the need to provide an enjoyable and seamless VR experience while staying true to the core 2048 gameplay, ensuring that the transition to the VR medium enhances rather than detracts from the player's engagement and satisfaction.

Description

- User Interface (UI) in VR: Designing an intuitive and visually appealing UI is paramount. In VR, this typically means creating interactive, 3D menus or HUD elements that users can manipulate with VR controllers. The UI should display essential information like the player's score, possible moves, and game instructions while avoiding clutter that can be overwhelming in a 3D environment.

- Controls and Interaction: The core 2048 gameplay elements, like swiping and merging tiles, should be reimagined for VR. Gesture-based controls that mimic the natural hand movements of users are essential. For instance, players can swipe their VR controllers to move tiles, physically touch or merge tiles together, and use spatial awareness to understand their position within the 3D grid. Implementing haptic feedback can provide sensory cues during these interactions.

- 3D Visuals and Immersion: The VR environment should be designed to immerse players in the 2048 world. This includes creating a 3D game board with lifelike tiles and background elements, ensuring that the depth and scale of the objects in the game feel realistic. Users should feel like they are standing amidst the 2048 grid. This immersion can be enhanced with dynamic lighting, animations, and particle effects.

- Comfort and Motion Sickness Mitigation: VR games must prioritize player comfort. Address motion sickness concerns by providing options for movement, like teleportation or smooth locomotion. Ensure that the frame rate is consistently high to avoid VR-induced nausea. Moreover, offering adjustable settings for players with varying levels of VR experience can greatly enhance the UX.

- Testing and User Feedback: Regular playtesting with VR users is essential to refine the UX. Gathering feedback about comfort, usability, and overall enjoyment can help identify and address potential issues and improve the game's design over time.

Module Description

In the realm of a UXVR project, crafting a 2048 game within Unity entails several key technical facets. This begins with configuring Unity for VR development, selecting the appropriate SDK or hardware ensuring a seamless VR setup. The development process centers on creating 3D models and assets, carefully optimizing textures and models to maintain a high frame rate crucial for VR. Gesture-based input is central to VR gameplay, often translating swipe and merge actions through VR controllers. The user interface takes on a 3D form, displaying scores, instructions, and hints in an immersive VR space. Game logic, adapted for VR, includes the movement and merging of tiles, all to align with VR user interactions. The 3D environment encompasses the game board, lighting, and particle effects, working together to engross players in the 2048 world. User comfort is paramount, achieved by introducing VR-specific options like teleportation or vignettes to combat motion sickness. Audio and visual feedback, including spatial audio, enhances the immersive experience. Rigorous testing and optimization are fundamental to maintain a high, consistent frame rate and ensure a comfortable experience in VR.

Brief Description of Software & Hardware Used and Its Programming

- Software:
  1. Unity Game Engine
  2. Adobe Illustrator
  3. Unity Asset Store

- Hardware:
  1. Storage
  2. Computer
  3. Mobile Device
  4. Keyboard

Code:

```
using System.Collections;

using TMPro;

using UnityEngine;

public class GameManager : MonoBehaviour{

    public TileBoard board;

    public CanvasGroup gameOver;

    public TextMeshProUGUI scoreText;

    public TextMeshProUGUI hiscoreText;

    private int score;

   private void Start(){

        NewGame()
 }

  public void NewGame()

    {

    // reset score
```

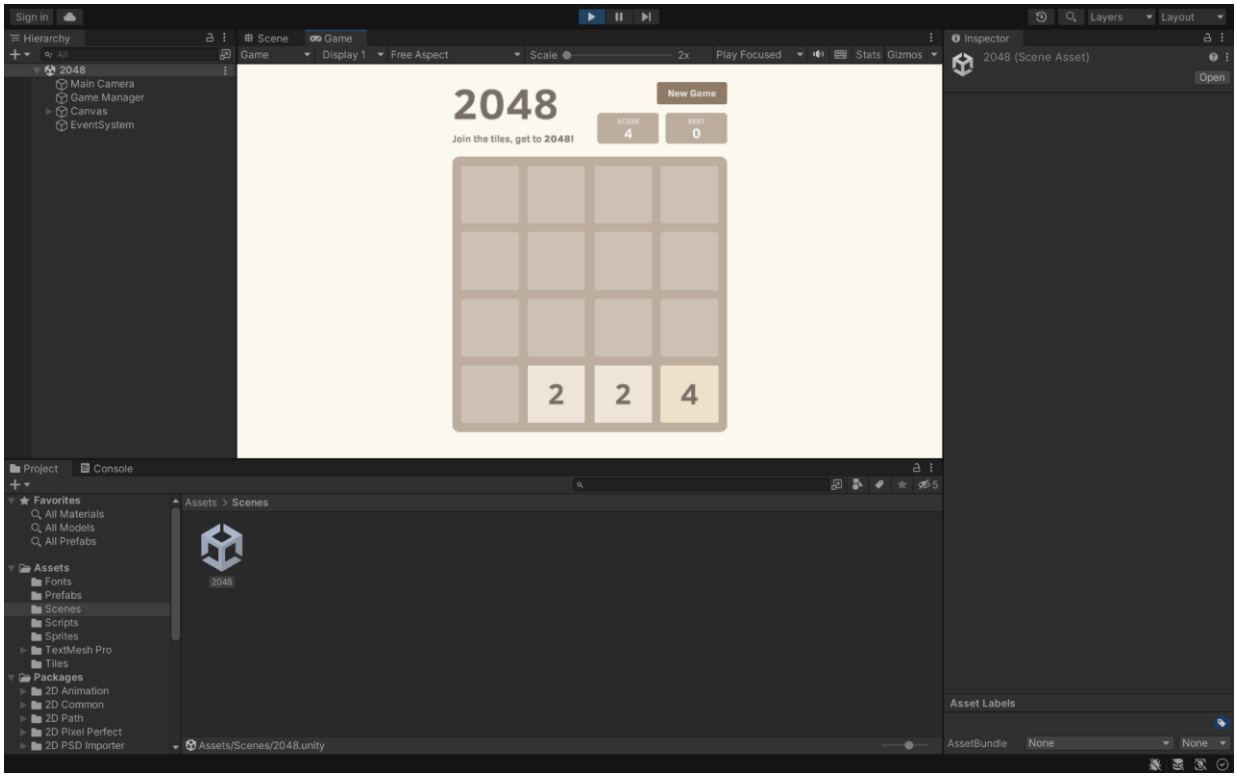
```
        SetScore(0);
        hiscoreText.text = LoadHiscore().ToString();
 // hide game over screen
        gameOver.alpha = 0f;
        gameOver.interactable = false;
        // update board state
        board.ClearBoard();
        board.CreateTile();
        board.CreateTile();
        board.enabled = true;
    }
 public void GameOver(){
        board.enabled = false;
        gameOver.interactable = true;
        StartCoroutine(Fade(gameOver, 1f, 1f));
    }
private IEnumerator Fade(CanvasGroup canvasGroup, float to, float delay = 0f){
        yield return new WaitForSeconds(delay);
       float elapsed = 0f;
        float duration = 0.5f;
        float from = canvasGroup.alpha;
while (elapsed < duration){
            canvasGroup.alpha = Mathf.Lerp(from, to, elapsed / duration);
```

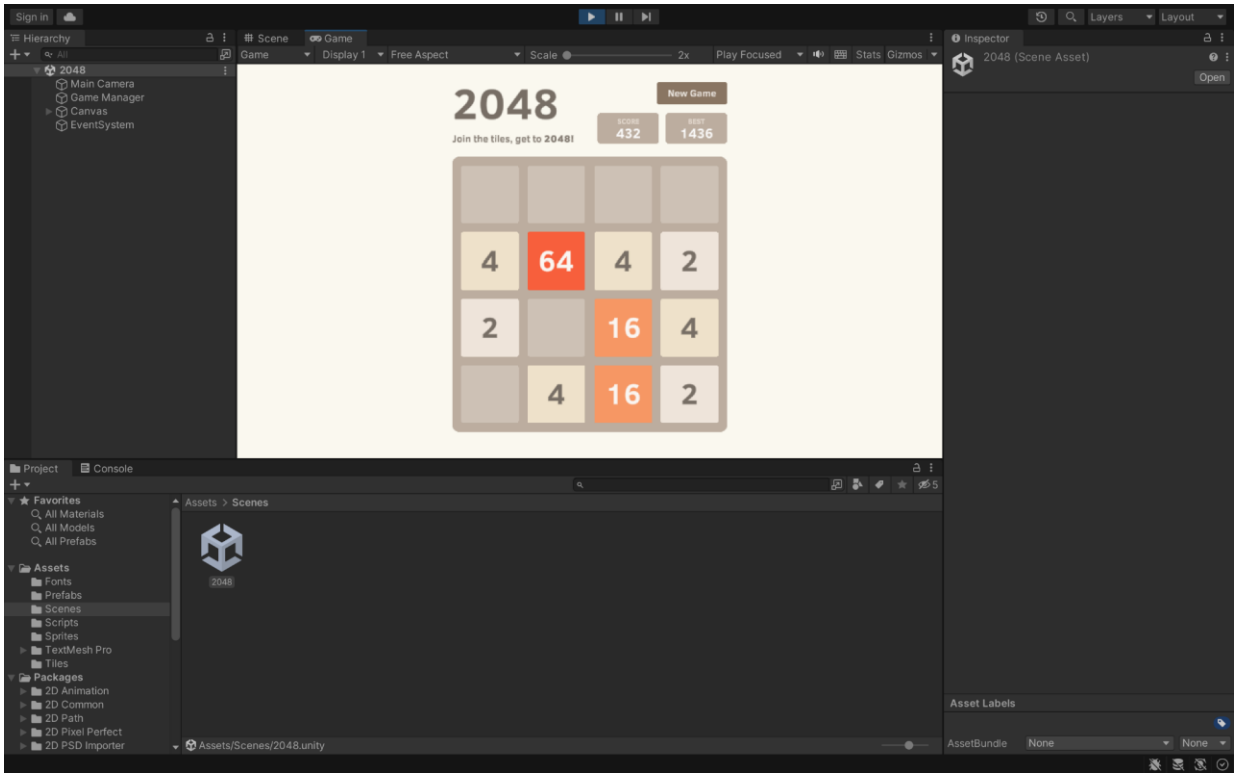
```
            elapsed += Time.deltaTime;
            yield return null;
        }
canvasGroup.alpha = to;
    }
public void IncreaseScore(int points){
        SetScore(score + points);
    }
private void SetScore(int score){
        this.score = score;
        scoreText.text = score.ToString();
          SaveHiscore()
          }
          private void SaveHiscore(){
        int hiscore = LoadHiscore();
          if (score > hiscore) {
            PlayerPrefs.SetInt("hiscore", score);
}
    }

    private int LoadHiscore(){
        return PlayerPrefs.GetInt("hiscore", 0);
}
```

}

Result:


2048
Join the tiles, get to 2048!
New Game
SCORE
4
BEST
0
2
2
4


2048
Join the tiles, get to 2048!
New Game
SCORE
432
BEST
1436
4
64
4
2
2
16
4
4
16
2

Conclusion

In conclusion, creating a 2048 game in Unity, with a focus on virtual reality (VR) requires a multi-faceted approach. It encompasses technical aspects like asset design, VR input integration, 3D environment creation, and performance optimization, all tailored to the VR platform. The development process also places significant emphasis on user comfort and engagement, with options to mitigate motion sickness and provide immersive feedback. Regular testing and user feedback play a vital role in fine-tuning the user experience.

References

[1] Juliani, Arthur, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. "Unity: A general platform for intelligent agents." arXiv preprint arXiv:1809.02627 (2018).

[2] Haas, John K. "A history of the unity game engine." (2014).

[3] Canossa, Alessandro. "Interview with nicholas francis and thomas hagen from unity technologies." In-Game Analytics, pp. 137-142. Springer, London, 2013.

[4] Becker-Asano, Christian, Felix Ruzzoli, Christoph Hölscher, and Bernhard Nebel. "A multi-agent system based on unity 4 for virtual perception and wayfinding." Transportation Research Procedia 2 (2014)

[5] Patil, Pratik P., and Ronald Alvares. "Cross-platform Application Development using Unity Game Engine." Int. J 3, no. 4 (2015).