Arnold Escobedo

Comp 565

Professor: Karamian

[Unity Project 1: Tetris]

Architecture/Design Document

Table of Contents

Version: <1.0>
Modifier: < Arnold Escobedo>
Date: 03/27/2019
Description of change: <Tetris Game Created>

---

1 Introduction

This document describes the architecture and design for the Tetris game developed at California state university Northridge. Tetris is a classic computer game developed by Alexey Pajitnov, and was completed in 1984. The player is given one of 7 tetriminos to place at the bottom of the grid or on top of other tetriminos. I, J , L, O, S, T or Z is spawned at the top of the grid and slowly moves towards the bottom. A random sequence of Tetriminos fall down the playing field. The objective of the game is to manipulate these Tetriminos, by moving each one sideways and/or rotating by quarter-turns, so that they form a solid horizontal line with no gaps. When such a line is formed, it disappears and any blocks above it fall down to fill the space. When a certain number of lines are cleared, the game enters a new level. As the game progresses, each level causes the Tetriminos to fall faster, and the game ends when the stack of Tetriminos reaches the top of the playing field and no new Tetriminos are able to enter. All of the Tetriminos can fill and clear both singles and doubles. Tetriminos I, J and L are able to clear triples. Only the I Tetromino has the ability to clear four lines simultaneously, and this is referred to as a "tetris". The scoring formula for the majority of Tetris games is built on the idea that more difficult line clears should be awarded more points. For example, a single line clear might be worth 100 points, while clearing four lines at once can be worth 800, and each subsequent back to back Tetris worth 1,200.

The purpose of this document is to describe the architecture and design of Tetris.

2 Design Goals and Objectives

Design priorities and objectives for the tetris game:

The game Tetris is created using the Unity game engine. We will need the following to start preparing the design and implementation for the game, to name a few:

- Modeling of the Tetriminos
- Game Logic
- User Input
- Scoring
- User Interface
- Music

3 System Behaviors

The system should display a 52 3-D cube tetrimino grid. When the player clicks the play button the first tetrimino should be spawned at the top of the grid. The player can move the tetrmino pieces left or right as long as they are within the tetrimino grid. The player can also change the way the tetrimino is position by pressing the up and down arrows. The player can make the tetrimino pieces fall faster by tapping the space button until the tetrimino piece reaches the next object. When the player creates a horizontal line the objects in that line will be destroyed and every object at the top will be shifted down. The player’s line score is then updated on the

UI so is their overall score. The level will be updated when the player clears 10 lines. The next tetrimino is displayed to the user at the bottom of the score UI. The tetris music for the game can be heard while the player is playing the game.

4 Logical Design and implementation

The system has 6 new classes and 3 new scenes;

Classes:

ITetrimino.cs , Menu.cs, UIManager.cs, GameManager.cs , Tetrimino.cs

Scenes:

Game , Gameover , Start

Details (Scenes):

The start scene displays the name of the game and a start button. When the user presses the start button the next scene will be the actual game. In the game scene, the user is playing the tetris game, with the score, level, lines, and next tetrimino information displayed. All tetris rules are applied in this scene. The game music is played in this scene and a nice background is displayed for the game. In this scene, the player can keep playing the game and every time they level up the next terimino will automatically increase its falling speed. When the player's tetriminos reach the top of the grid the gameover scene will be displayed and the user will have the option of playing the game again.

Details (Classes):

The ITetrimino class gets the current tetrimino falling type and handles the methods that will push the teriminos down (PushDown()), check whether they are within the grid(CheckBoard()), get the rotation (GetRotation()), pivot (GetPivot()) and root(GetRoot()). The Menu.cs class handles resetting the UI Scores; currentScore(), currentLevel(), NumLines(). The UIManager class handles all of the UI information such as the current score, level, and number of lines cleared. Inside the Update() function the UIInformation() method is called. This method will update Score.text, Level.text, and Lines.text. The GameManager class handles the logic of the game such as the random spawning tetriminos, the preview tetrimino, fall, points, score, level, and lines cleared. The UpdateScore() method checks how many lines we're cleared if the number of rows in turn is equal to zero. ClearedOneLine(), ClearedTwoLines(), etc methods will update the score whenever the player clears a certain amount of lines. To spawn the next tetrimino spawnNextTetrimino() is called within the start function. Inside the update function we call UpdateScore, updateLevel(), and updateSpeed() to constantly be checking for any changes in the game. The function updateLevel() will update the current level when the player clears 10 lines. updateSpeed() updates the speed when the user clears 10 lines. The method check is AboveGrid() will check whether the tetrimino goes above the grid if it does then "true" will be returned and the game will be over. The method IsSFullRowAt() will check when the player gets a full row and will update the score. The method moveRowDown() will accept a row at position y and individually move rows dows when moveAllRowsDown() is called. The method DeleteRow() will delete a row when the user has filled up the horizontal axis. insideGrid() will check the width and height of the grid to make sure that the tetriminos are confined within the grid. The tetrimino class controls and detects what buttons the player is pressing and also checks whether each tetrimino is at a valid position.

5 User Interactions

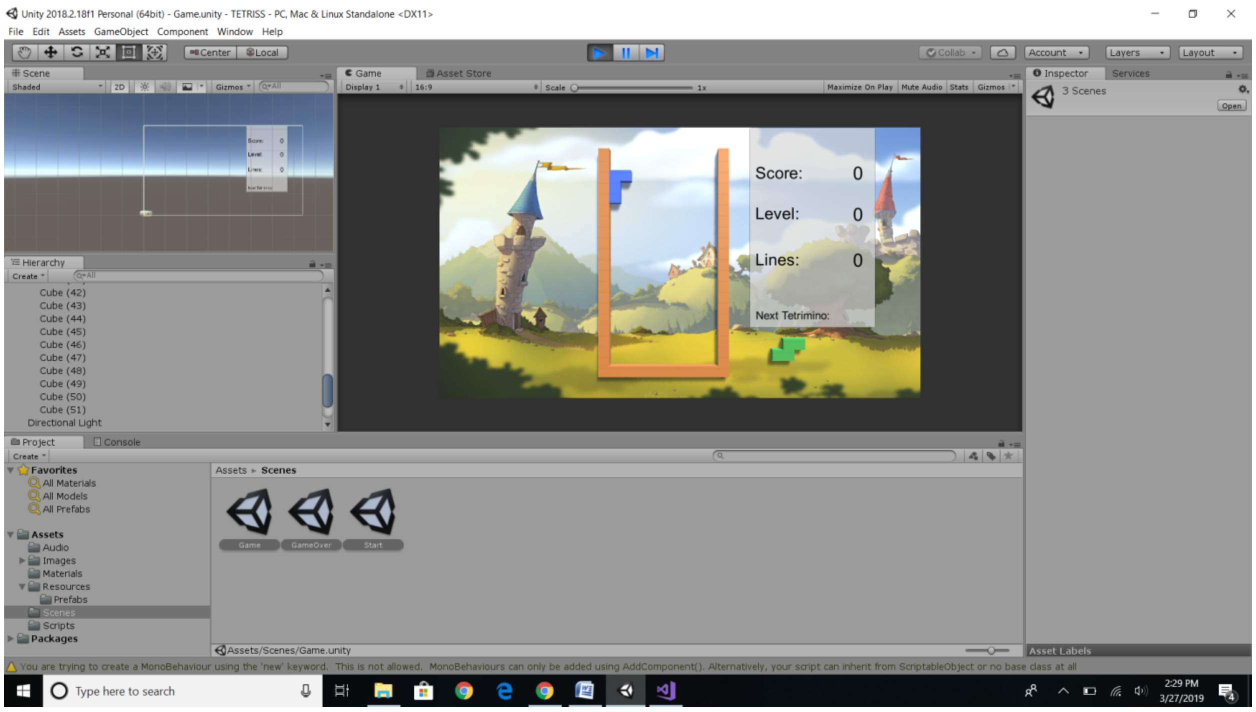
The start scene is displayed to the player when the player clicks on the start button they are sent to the game scene where they can begin to play tetris. In the game all of the game controls can be performed by the player. When the player presses the left or right arrows they can move the tetrimino left or right. When they press the up or down arrows the player can change the angle of the tetrimino. The tetrimino will be confined within the grid so if the player tries to move out they will not be able to. When the player presses the space bar the tetrimino piece will increase its fall speed. If the user fills the grid with tetrimino pieces and fills it up to the top the game will be over. The player's score, lines, and current level will be updated while the player is playing the game.

6 User Interface

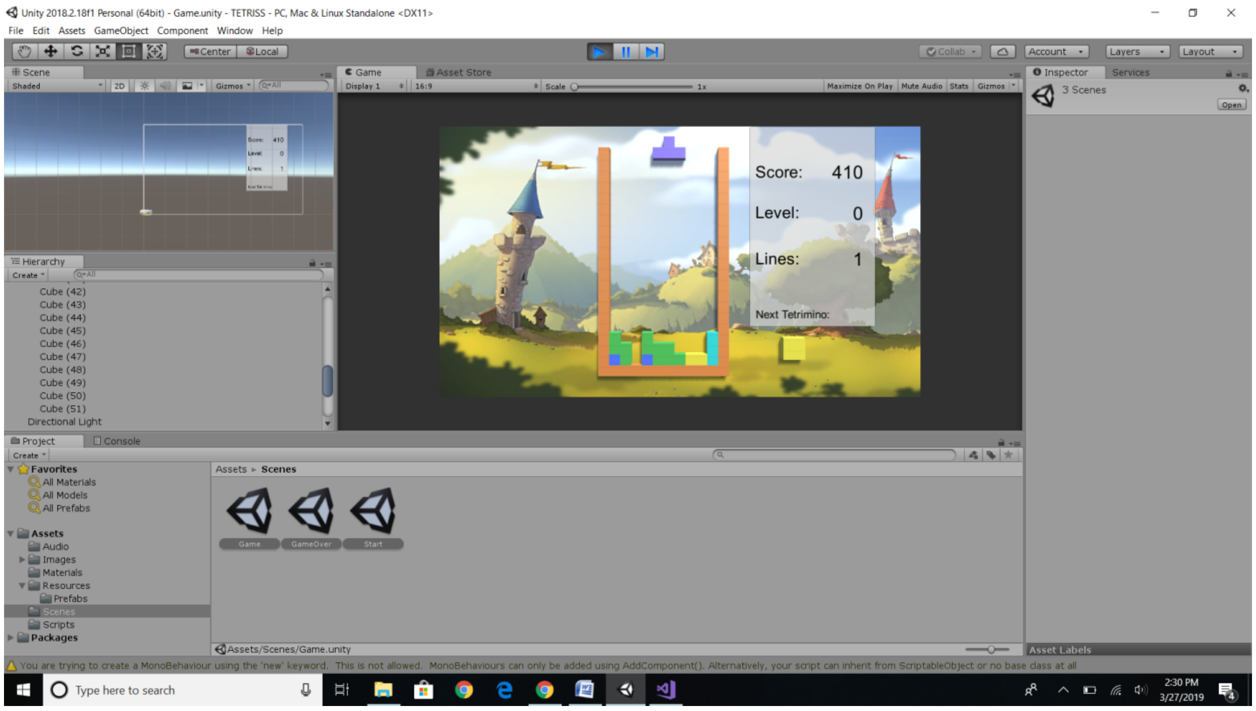
The user interface simply includes the following:

-The player's score
-The number of lines that the player has cleared
-The next tetrimino
-The current level that the player is in

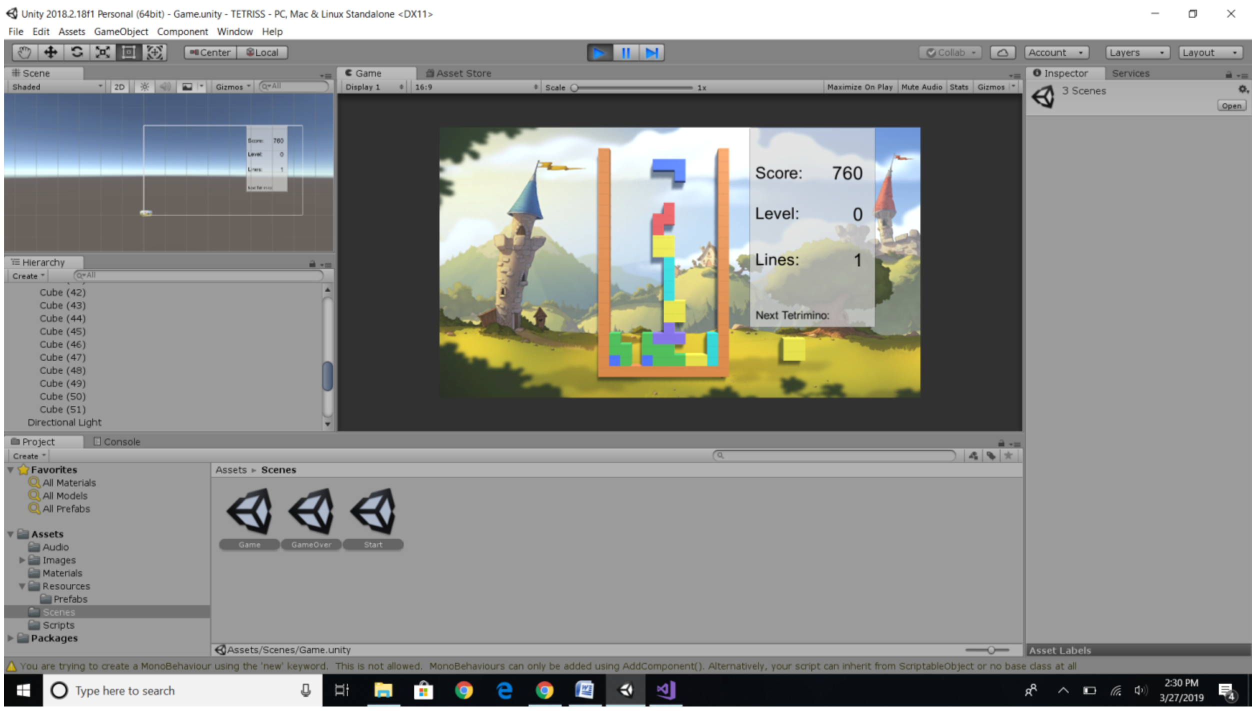
7 Screenshot


Unity 2018.2.18f1 Personal (64bit) - Game.unity - TETRISS - PC, Mac & Linux Standalone <DX11>
Score: 0
Level: 0
Lines: 0
Next Tetrimino:

You are trying to create a MonoBehaviour using the 'new' keyword. This is not allowed. MonoBehaviours can only be added using AddComponent(). Alternatively, your script can inherit from ScriptableObject or no base class at all

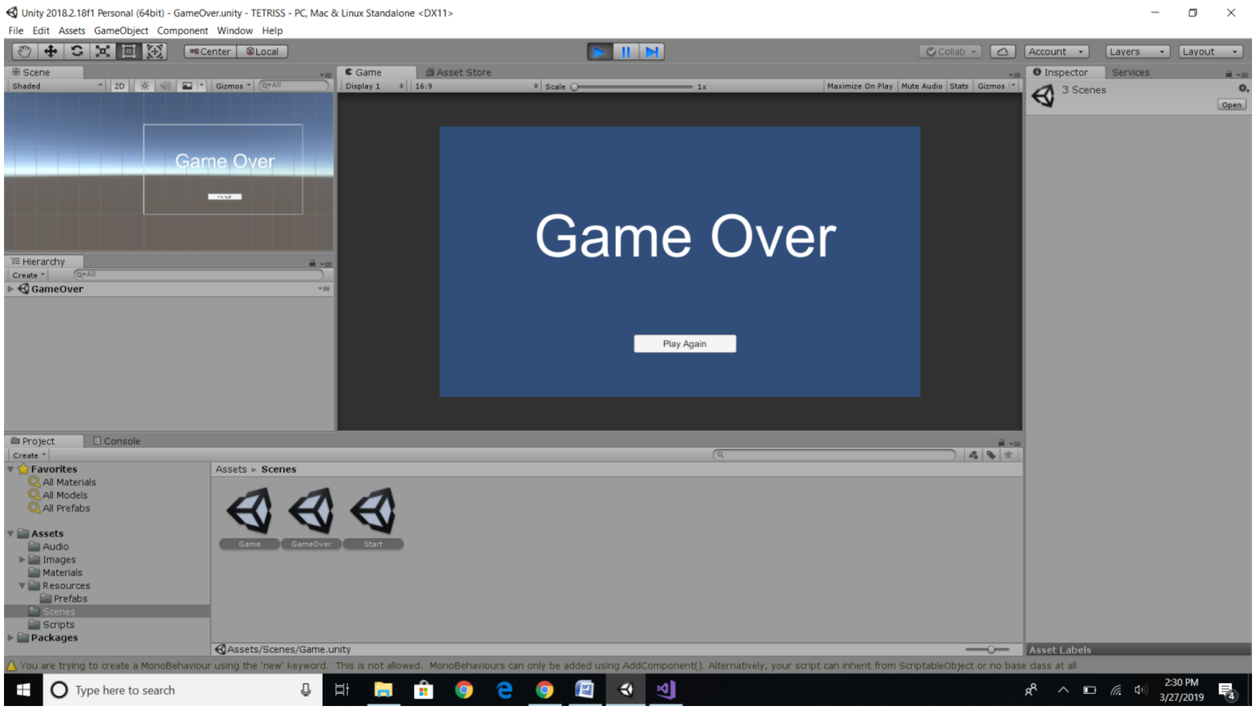

Unity 2018.2.18f1 Personal (64bit) - Game.unity - TETRISS - PC, Mac & Linux Standalone <DX11>
Score: 410
Level: 0
Lines: 1
Next Tetrimino:

You are trying to create a MonoBehaviour using the 'new' keyword. This is not allowed. MonoBehaviours can only be added using AddComponent(). Alternatively, your script can inherit from ScriptableObject or no base class at all


Unity 2018.2.18f1 Personal (64bit) - Game.unity - TETRISS - PC, Mac & Linux Standalone <DX11>

Assets/Scenes/Game.unity

⚠ You are trying to create a MonoBehaviour using the 'new' keyword. This is not allowed. MonoBehaviours can only be added using AddComponent(). Alternatively, your script can inherit from ScriptableObject or no base class at all


Unity 2018.2.18f1 Personal (64bit) - GameOver.unity - TETRISS - PC, Mac & Linux Standalone <DX11>

Assets/Scenes/Game.unity

⚠ You are trying to create a MonoBehaviour using the 'new' keyword. This is not allowed. MonoBehaviours can only be added using AddComponent(). Alternatively, your script can inherit from ScriptableObject or no base class at all

8 Lessons Learned

For this tetris project we learned about prefabs and how to create pivot points. We also learned about collision detection like how to stop a tetrimino from going outside the grid and how to control the tetrimino pieces with arrows and space bars. Also how to detect when a horizontal line is filled so that the player's score can be updated. I learned how to add color, music, and a background to the game. How to create and handle a UI for the game was difficult. I learned how to have the user navigate from scene to scene within the game. Finally, I learned how to spawn a game object at a certain position in the game.

9 Summary

Creating the classic tetris game was very difficult. The most difficult part was detecting collisions and also making sure that every game object was in the correct position so that the objects wouldn't overlap other objects. Deleting rows wasn't that difficult since all I had to do was pass each individual object to be destroyed when the game detected a full row. Implementing the user controls wasn't too difficult but I did have to make sure that the empty game object position controlling the prefab was in the correct position so that the tetrimino piece wouldn't move up, left, down, or right when switching angles. Adding music and background was simple.