The Impossible Game

Franklin Novak
franklin.novak001@umb.edu
University of Massachusetts Boston

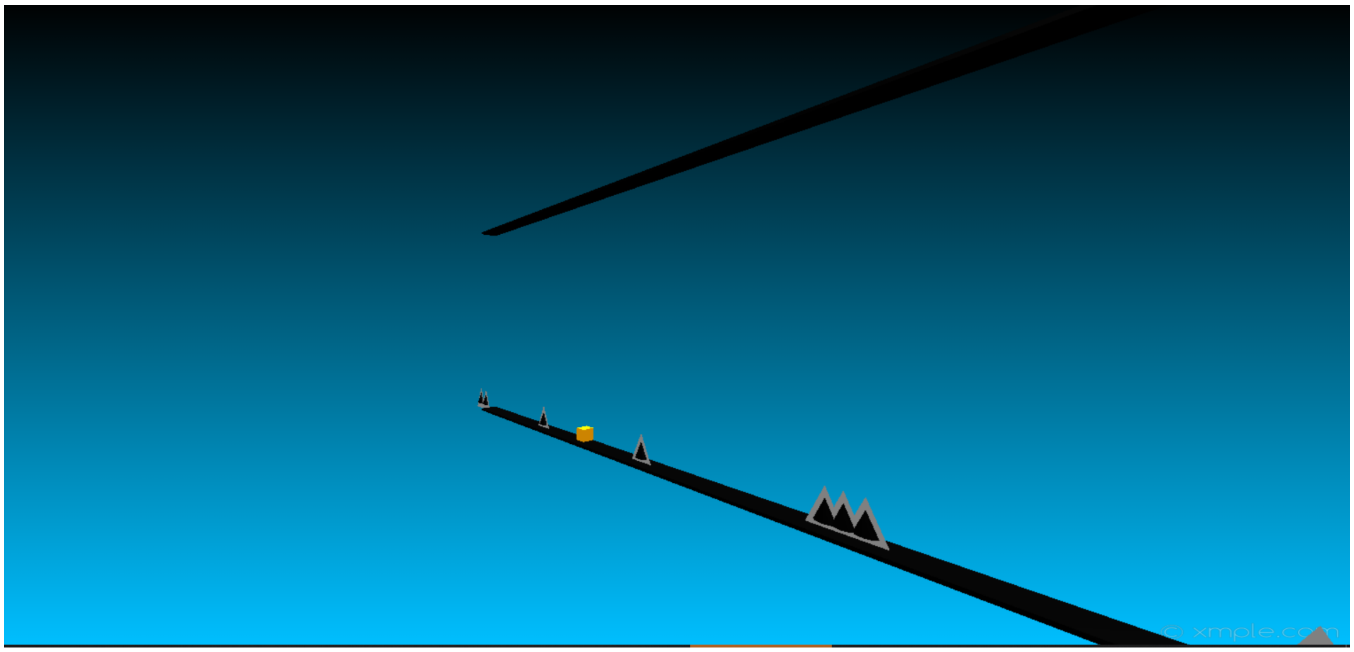

Figure 1: 3D Impossible Game!

ABSTRACT

This projects was aimed to create a 3D rendition of the popular game, the Impossible Game. In this game the player controls an orange cube and must maneuver a course of obstacles in its way. If the player runs into one of the obstacles they will have to restart the game. The player can use space bar and the up and down arrow keys to jump and traverse over the obstacles coming their way. The camera position will shift as the player makes it further and further through the level.

KEYWORDS

WebGL, Visualization, JavaScript, Object Oriented Programming, Collision Detection

ACM Reference Format:
Franklin Novak. 2019. The Impossible Game. In *CS460: Computer Graphics at UMass Boston, Fall 2019.* Boston, MA, USA, 3 pages. https://CS460.org

1 INTRODUCTION

This project was very important to me personally. I have always wanted to create video games for people to play and I saw this as a great opportunity to do just this. Since this project is run on a website anyone with internet connection will be able to enjoy it. I worked on this project alone so all contributions were mine with the help of https://threejs.org/ for tips on implementation.

2 RELATED WORK

I would like to cite Three.js [1].

3 METHOD

This project uses Three.js, javascript, and html to create a 3D version of The Impossible Game. The player controls a cube rendered using the THREE.BoxGeometry from Three.js. This cube is controlled using the space bar and the up and down arrow keys. There are obstacles that are rendered using the THREE.ConeGeometry from Three.js. These obstacles slide towards the player and they must jump over them. The jumping of the cube is just simple javascript but achieves the desired function. After the player has made it through the first section of the level gravity is inverted and the player now tries to avoid obstacles while on the roof. After this section the player is then lowered down into the middle of the level and must use the arrow keys to move up and down to avoid walls coming at them. These walls are again rendered using the THREE.BoxGeometry. Once the player has successfully navigated the level they will see a red and white checkered wall coming at them signifying the completion of the level.

3.1 Implementation

I used mostly Three.js to complete this project. All of the obstacles, the floor and roof, and the player are all rendered using Three.js. To get the side scrolling illusion instead of having the player move the obstacles move towards the player. To create all of the obstacles I used object oriented programming. By doing this I was able to use variables I created for each obstacle. One of these variables was the velocity in the X direction. I used this to move all the obstacles at a consistent speed.

```
  objects[i].obstacle_x_velocity -= .025
  objects[i].outline_x_velocity -= .025;
 objects[i].obstacle.position.x += objects[i].obstacle_x_velocity
```

Next I implemented a framecounter that updated each frame. When the framecounter variable reached certain thresholds that I determined the camera position would change therefore taking advantage of the fact that the scene was 3D.

```
    if(framecounter > 150){
        camera.position.x = 100
     }
```

The last thing that was necessary to complete the game was collision detection. I was able to do this by using Box3 objects from Three.js. If a collision was detected than the page would refresh therefore restarting the game.

```
     var collision = boxBB.intersectsBox(spikeBB);
        if(collision == true){
         refreshPage();
        }
```

3.2 Milestones

3.2.1 Milestone 1. My first milestone was to decide what my project would be. I really enjoyed assignment 4 so I decided I wanted to create a new version of the Impossible Game.

3.2.2 Milestone 2. My second milestone was creating gravity within the scene so that the block could jump and come back to the surface.

3.2.3 Milestone 3. My third milestone was using object oriented programming to get all of the obstacles I wanted to create to move.

3.2.4 Milestone 4. My fourth milestone was inverting gravity and creating the obstacles on the roof instead of the floor.

3.2.5 Milestone 5. My fifth milestone was implementing collision detection so that the page would refresh automatically when the player lost.

3.2.6 Milestone 6. My sixth milestone was implementing a framecounter to use to move the camera around at different points.

3.2.7 Milestone 7. My seventh and final milestone was creating the red and white striped line so the player knows when they have completed the level.

Table 1: Performance when obstacles are removed from scene or left in

Remove or Leave	Performance
Remove	60 FPS
Leave	24 FPS

3.3 Challenges

- Challenge 1: Getting all obstacles in scene to move at a constant speed since it is based off of distance from the player.
- Challenge 2: Collision detection took quite some time to figure out.
- Challenge 3: Implementing gravity, anti-gravity, and flying.
- Challenge 4: Making the level beatable. It is very challenging and probably took me hundreds of tries to complete but it is doable.
- Challenge 5: Finding the camera angles I wanted based on the framecounter.

4 RESULTS

I am very proud of my final result. The game is extremely challenging yet addicting. You will get to a certain point in the level and lose then struggle to reach that same spot for many attempts. At first the player slides across the floor jumping over obstacles but then once that section is passed they will float up to the ceiling and have to play the game upside down. If the player can get past this point they will begin to fly and have to go over and under walls coming their way. This section of the game requires precise timing or the player will make contact with the wall causing them to have to restart from the beginning.

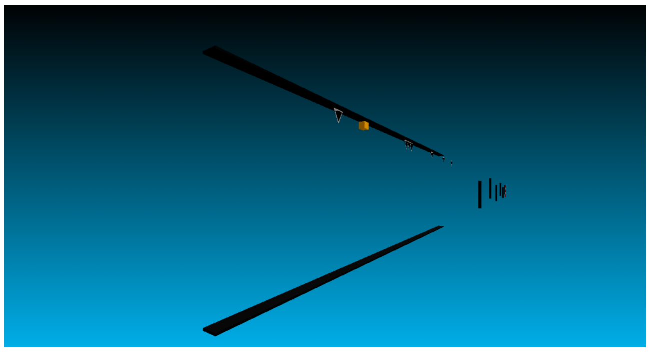

Figure 2: The second section of the game where gravity is inverted

5 CONCLUSIONS

Overall this project was extremely fun to complete. While at times it was extremely frustrating the knowledge I gained was very valuable. I now understand how to do things such as adding gravity to a scene

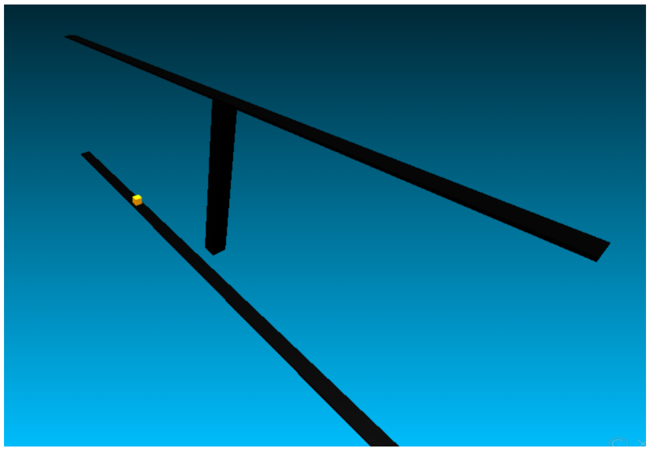

Figure 3: The third and final section where player must avoid walls.

and collision detection between 2 different objects. I also gained experience in another area of game design that is very important. This area was level design. I had to first think about how to create the level, second I had to make sure that the level was balanced and beatable. It was really interesting to connect so many of the different things that we learned this semester in class. I also enjoyed how the final project was a creative choice. This made it so that I really wanted to work on the project to make it as fun as I could. All of the projects we created had me more than prepared to take on the challenge of creating this game.

I think the aspect of the project I am most proud of in the object oriented programming. I was having extreme difficulty getting everything to move together then I remembered our robots that would all dance in unison. I went back and studied this code to understand how it worked and how I could implement it into the game. Once I got it working I had a real feeling of satisfaction that I was able to connect all of these things. The game is very hard but very fun at the same time. I had some of my friends try it out and so far I am the only one who can beat the game. I had doubts that it was possible but after about an hour of trying I was finally able to complete the game. I am very happy that I now have the knowledge of Three.js to make games and levels such as these. I plan on adding move levels and new kinds of obstacles over the break from school and into the foreseeable future!

REFERENCES

[1] Ricardo Cabello et al. 2010. Three.js. *URL: https://github. com/mrdoob/three.js* (2010).