COLLABORATIVE PROJECT

A final project submitted to the University of Wales in partial fulfilment of the requirements of BA (Hons) Digital Media in Game Development

June 2013

MagnoBall

Pavankumar Chopra

2nd Year UG – Game Development

11UG03040

ICAT, Design and Media College,

Bangalore

Contents

High Concept

One Liner:

Playing responsible and safe Karl has to obey and follow the traffic rules and regulations of the road. Apart from following traffic rules Karl has to avoid hitting the upcoming traffic or obstacles that will make Karl break traffic rules. The longer Karl can stay on road without breaking the traffic rules the higher the score he will get.

Core Tenets:

Cru zing along the traffic and following the rules of the road. Karl must also keep a look out for traffic cops and always maintain an appropriate speed. To do so he has the ability to brake and control his speed but has to also make sure he has enough fuel to catch up speed.

- Speed Check Posts
- All Green Signals
- Time Wrap
- Traffic cross roads
- Rail way impact

Game Genre

This is an Arcade Driving Game.

Re-playability

The games re-playability depends on the player. He can play the game again if he or she isn't satisfied with their score and wants to get a higher score in-order to beat their friends score. A player would also want to play the game again if she or he likes the UI, the games audio or even the visual appeal of the game. The Player can collect coins to buy new vehicles to try out and use in game. All these options give the player a reason to play again and thus making the game re-playable.

Target Audience

The game is targeted to players of the age 13 and above. The games adventurous, challenging and is having a very cartoonish appeal. The game is a lot fun and interesting and challenges players to try getting a top score. Although the game targeted for teens and adults, the game provides a learning experience for everyone to drive safe and follow rules on the road. It also teaches us what happens when we do not follow rules and the penalty we face.

Game Play

Karl is found cru zing on a busy road. The player plays Karl and helps him avoid the incoming traffic and follows the rules of Road. By playing safe and keeping the rules in mind the player has to try traveling the longest distance possible and getting a high score. For ever traffic rule Karl breaks his score will be misused. He will also have to collect fuel along the road to fill his fuel bar so that he doesn't stop on the way. If Karl runs out of fuel and stops, cars behind him will crash into him.

Gameplay & Game Mechanics:

The gameplay of the game is very simple. The player has to avoid the floating cars and other obstacles. It is an endless running game. The player has swipe option to change lanes and avoid obstacles.

The game mechanics are the swipe controls for the character movement and top scroll for changing the gravity from positive to negative the either way around.

Character Description:

Name:

Zlosh

Description:

The character is in a futuristic world with all the flying cars with the help of magnetic field. The character has 3 major parts in him which look like tyre. There is a main tyre and two small tyre helping it for directions. These tyres are connected with a flexible rods. The character is also floating with magnetic field.

Modelling Description:

The character is modelled with the lowest Polly on the main tyres and a small rod type keeping them in contact with all the tyres.

Animation Description:

The animation cycles required for this games are idle cycle where the character is just still with his tyre rotating. There is a move cycle where the character bends slightly towards the sides and comes back to normal position. Flip animation where the character turns totally upside down. These are the animations required.

Level Description:

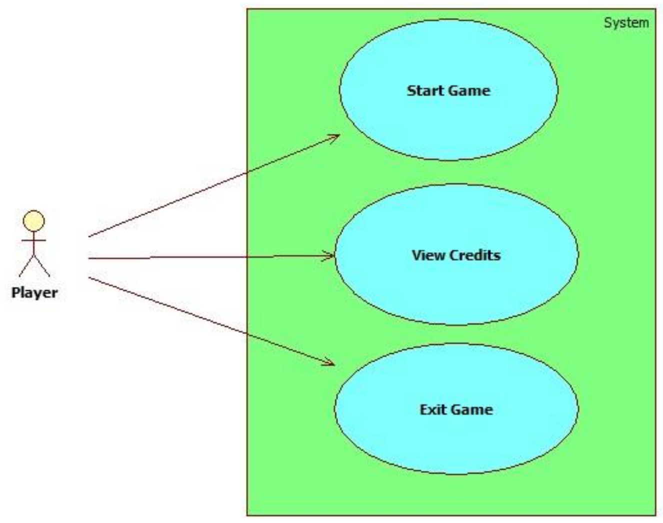
The level has an unending track with some obstructers on the path like the floating car and light lamp. The player is made to stay still in the level and the prefabs and other models are made to move in the level. For the repetition of the path some Java scripts and C# scripts are used. The obstacles are placed properly with proper spacing so that the character has some place to stand. . The score system in the level is quite simple as long as the player tries to stay alive his points go on increasing. The sore is calculated on the distance and pick-ups.

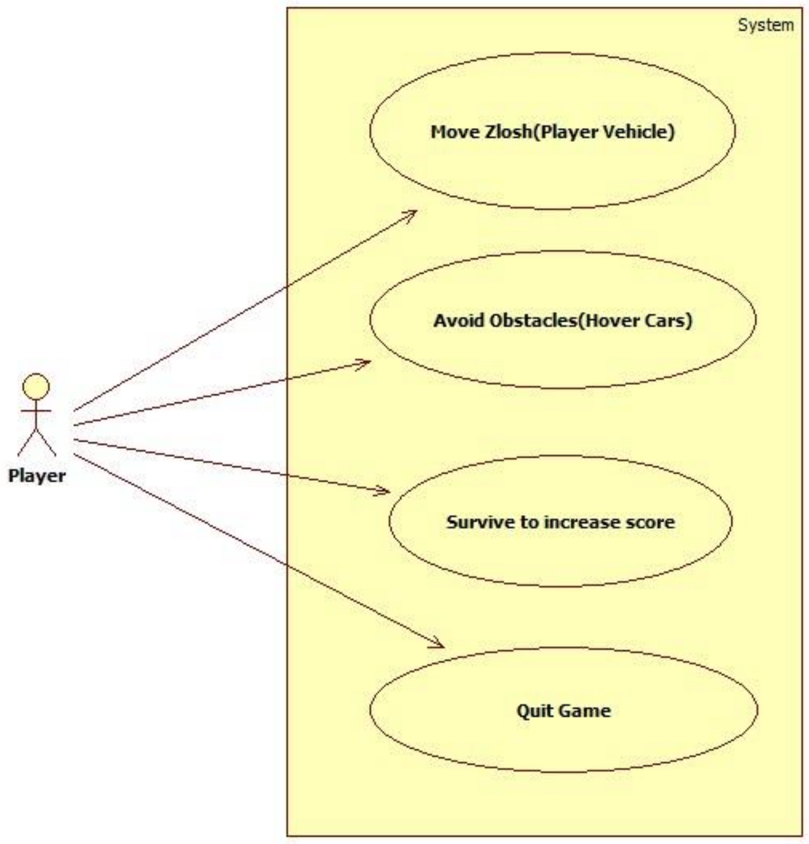
UML DIAGRAMS

USE CASE DIAGRAM

Description: The below diagram represents the actions, user can performed in the game. This diagram helps us to know all the actions user can perform in the game.

Diagram:

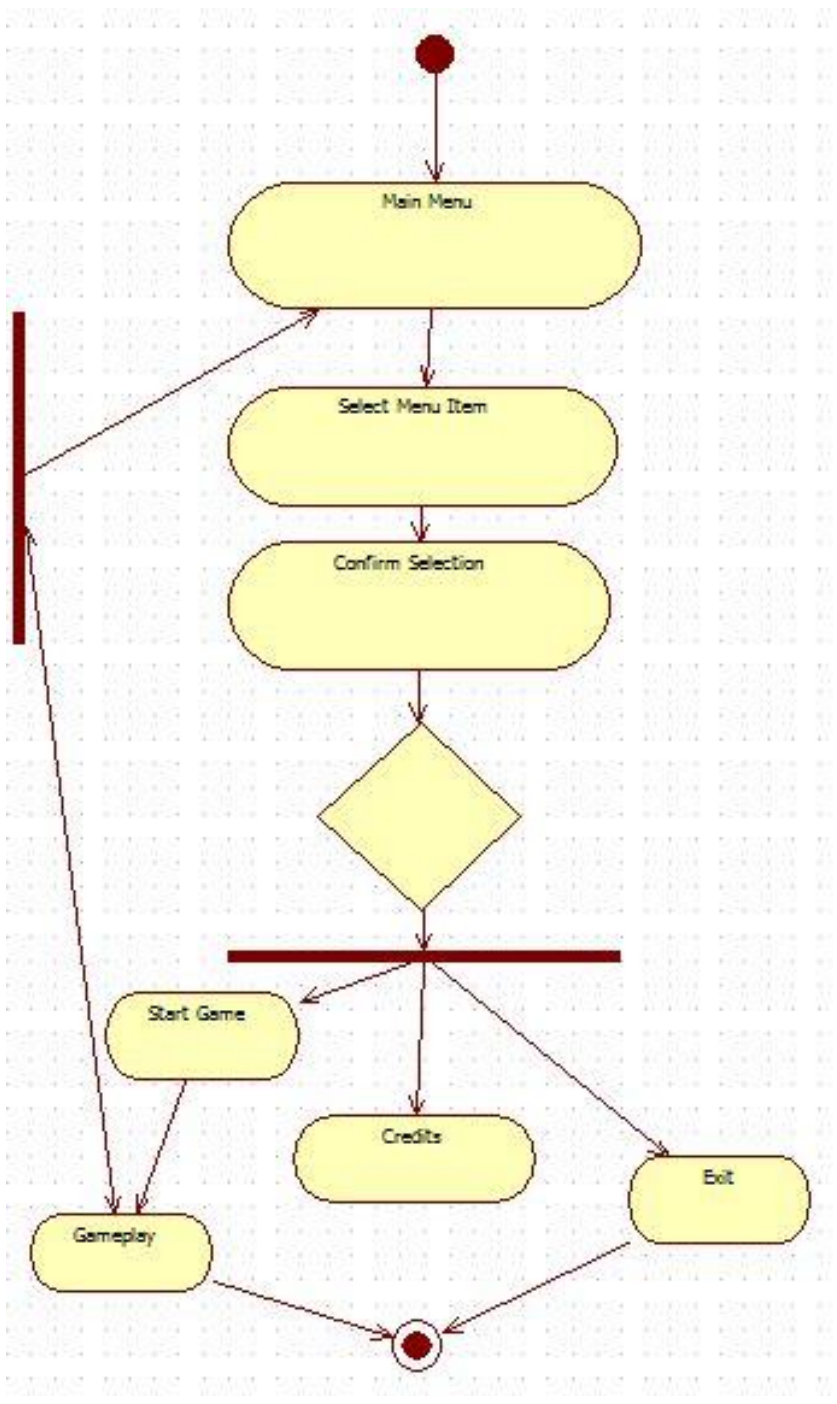



System
Move Zlosh(Player Vehicle)
Avoid Obstacles(Hover Cars)
Survive to increase score
Quit Game
Player

ACTIVITY DIAGRAM

Description: Activity Diagram is used to show the game-play mechanics working parallel.

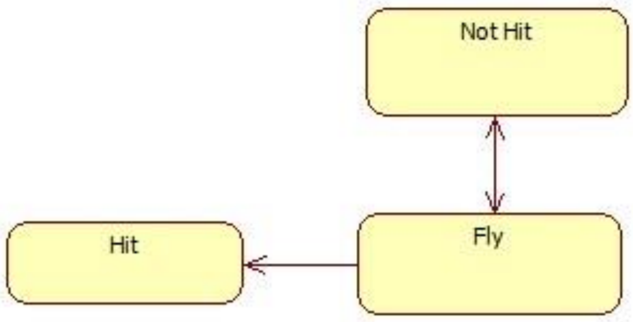
Diagram:



STATE DIAGRAM

Description: The diagram represents States of the Player. Activity Diagram helps us to know what activities can be performed by a Player or an AI.

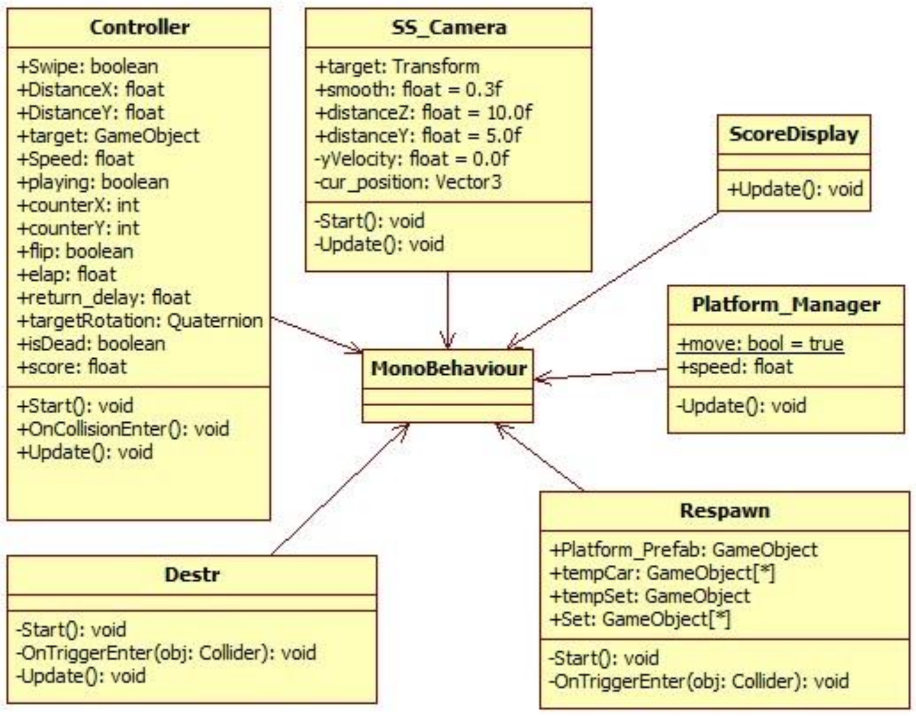
Diagram:



CLASS DIAGRAM

Description: The below diagram represents the class diagram. Then some important variables in the class and then functions declared in the class.

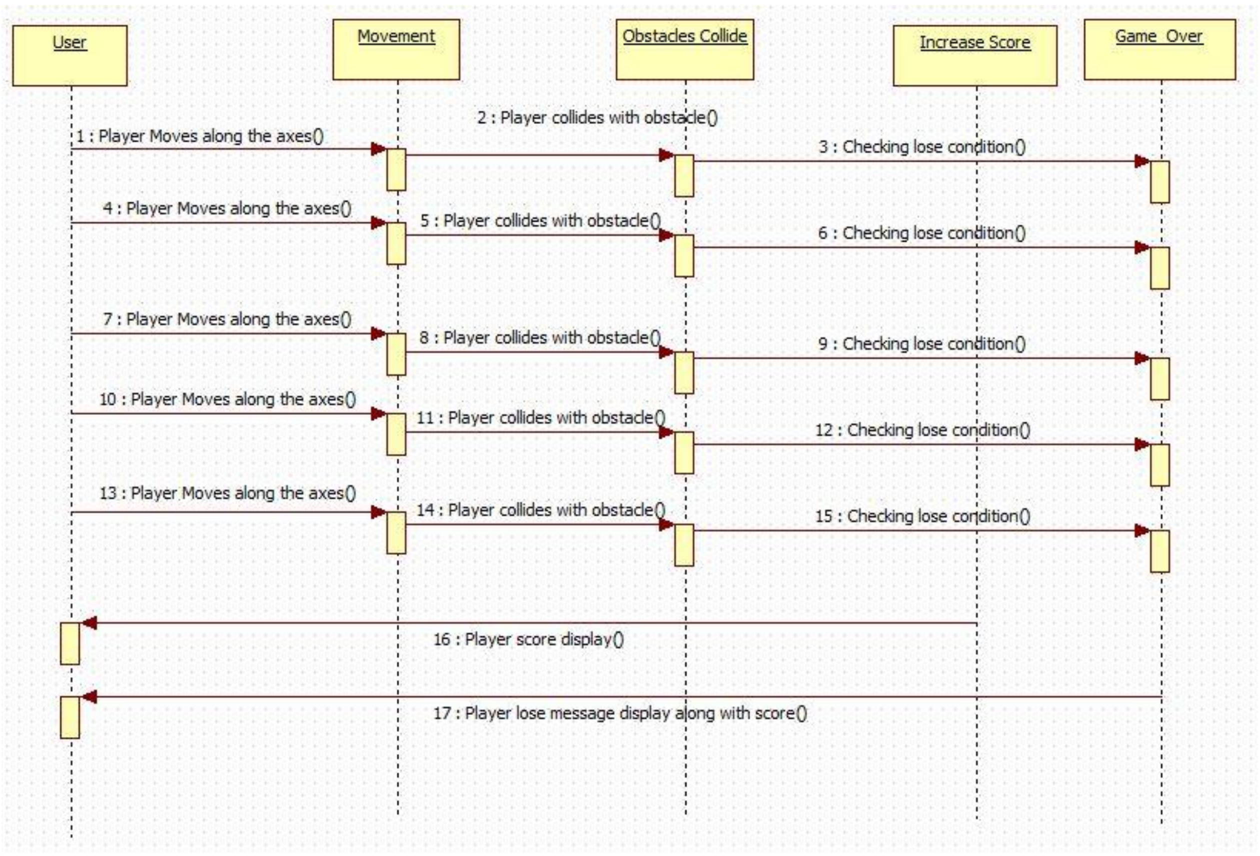
Diagram:



SEQUENCE DIAGRAM

Description: Sequence Diagram helps user to know the interaction between Player and the varies Puzzles in the game.

Diagram:


User
Movement
Obstacles Collide
Increase Score
Game_Over
1 : Player Moves along the axes()
2 : Player collides with obstacle()
3 : Checking lose condition()
4 : Player Moves along the axes()
5 : Player collides with obstacle()
6 : Checking lose condition()
7 : Player Moves along the axes()
8 : Player collides with obstacle()
9 : Checking lose condition()
10 : Player Moves along the axes()
11 : Player collides with obstacle()
12 : Checking lose condition()
13 : Player Moves along the axes()
14 : Player collides with obstacle()
15 : Checking lose condition()
16 : Player score display()
17 : Player lose message display along with score()

SOURCE CODE

SS_Camera.cs

```
public class SS_Camera : MonoBehaviour {

    public Transform target;
    public float smooth = 0.3f;
    public float distanceZ = 10.0f;
       public float distanceY = 5.0f;
    private float yVelocity = 0.0f;
       Vector3 cur_position;
       // Use this for initialization
       void Start () {
         Vector3 cur_position = new Vector3 (target.position.x, distanceY, distanceZ);
       }

       // Update is called once per frame

    void Update()
       {
              if(target==null)
                     return;

        float yAngle = Mathf.SmoothDampAngle(transform.eulerAngles.y,
target.eulerAngles.y, ref yVelocity, smooth);
        cur_position = new Vector3 (cur_position.x, distanceY, target.position.z);
        cur_position += new Vector3(0.0f, 0.0f, -distanceZ);
        transform.position = cur_position;
        transform.LookAt(target);
    }
}
```

Controller.js

```
var Swipe : boolean;
var DistanceX : float;   //0.675
var DistanceY : float;
var target : GameObject;
var Speed : float;
var playing : boolean;
var counterX : int;
var counterY : int;
var flip : boolean;
var elap : float;
var return_delay : float;
var targetRotation : Quaternion;
static var isDead : boolean;
static var score : float;

function Start () {
counterX = 0;
counterY = 0;
targetRotation = Quaternion.identity;
score = 0;
isDead = false;
}

function OnCollisionEnter(coll : Collision)
```

```
{
if(coll.gameObject.tag == "Collide")
       isDead = true;
}

function Update () {

if(animation.IsPlaying("Right") || animation.IsPlaying("Left"))
       playing = true;
else
       playing = false;


if(!playing)
       animation.Play("Idle");

if(this.counterX == 2 || this.counterX == -2)
{

elap += Time.deltaTime;
       if(this.elap > return_delay)
       {
              this.elap = 0;
              if(counterX == 2)
              {
              this.target.transform.position -= Vector3(this.DistanceX,0,0);
              counterX-=1;
              }
              if(counterX == -2)
              {
              this.target.transform.position += Vector3(this.DistanceX,0,0);
              counterX+=1;
              }
       }
}
else
       this.elap = 0;

this.transform.position = Vector3.Lerp(this.transform.position,
this.target.transform.position, Speed*Time.deltaTime);
if(this.flip)
{
       this.transform.rotation = Quaternion.RotateTowards(this.transform.rotation,
this.targetRotation,500*Time.deltaTime);
}

//PC Controls
if(!playing)
{
       if(Input.GetKeyDown(KeyCode.D) && counterX < 2)
       {
              this.target.transform.position += Vector3(this.DistanceX,0,0);
              animation.CrossFade("Right");
              counterX += 1;
       }

       if(Input.GetKeyDown(KeyCode.A) && counterX > -2)
       {
              this.target.transform.position -= Vector3(this.DistanceX,0,0);
              animation.CrossFade("Left");
              counterX -= 1;
       }
```

```
		if(Input.GetKeyDown(KeyCode.W) && counterY < 1)
		{
			this.target.transform.position += Vector3(0,this.DistanceY,0);
			this.targetRotation *= Quaternion.Euler(0,0,180);
			flip = true;
			counterY += 1;
		}
		if(Input.GetKeyDown(KeyCode.S) && counterY > 0)
		{
			this.target.transform.position -= Vector3(0,this.DistanceY,0);
			this.targetRotation *= Quaternion.Euler(0,0,180);
			flip = true;
			counterY -= 1;
		}
	}

	//Android Touch/Swipe Controls
	if(Input.touchCount > 0)
	{
		var touch : Touch;
		touch = Input.GetTouch(0);
		if(touch.phase == TouchPhase.Began)
			Swipe = true;
		if(touch.phase == TouchPhase.Moved)
		{
			if(this.Swipe)
			{
				if(touch.deltaPosition.x > 10 && counterX < 2)
				{
					this.target.transform.position += 
	Vector3(this.DistanceX,0,0);
					animation.CrossFade("Right");
					counterX += 1;
					Swipe = false;
				}

				if(touch.deltaPosition.x < -10 && counterX > -2)
				{
					this.target.transform.position -= 
	Vector3(this.DistanceX,0,0);
					animation.CrossFade("Left");
					Swipe = false;
					counterX -= 1;
				}

				if(touch.deltaPosition.y > 10 && counterY < 1)
				{
					this.target.transform.position += 
	Vector3(0,this.DistanceY,0);
					this.targetRotation *= Quaternion.Euler(0,0,180);
					flip = true;
					counterY += 1;
					Swipe = false;
				}

				if(touch.deltaPosition.y < -10 && counterY > 0)
				{
					this.target.transform.position -= 
	Vector3(0,this.DistanceY,0);
					this.targetRotation *= Quaternion.Euler(0,0,180);
					flip = true;
```

```
                        counterY -= 1;
                        Swipe = false;
                    }
                }
            }
            if(touch.phase == TouchPhase.Ended)
                Swipe = false;
    }
}
```

Destr.cs

```
public class Destr : MonoBehaviour {

	// Use this for initialization
	void Start () {
	
	}
	void OnTriggerEnter(Collider obj)
	{
		Debug.Log("Destroy Triggerd");
		if(obj.gameObject.tag == "Player")
		{
			// Destroy the Prefab
			Debug.Log("Destroyed");
			DestroyObject(this.transform.parent.gameObject);
		}
	}
	// Update is called once per frame
	void Update () {
	
	}
}
```

Display.js

```
var delay : float;
var Dead_Delay : float;
private var Dead_elapsed : float;
private var elapsed : float;

function Start () {

}

function Update () {

elapsed += Time.deltaTime;
this.guiText.text = ""+Controller.score;
if(this.elapsed > delay && !Controller.isDead)
{
	elapsed = 0;
	Controller.score++;
}

if(Controller.isDead)
{
	Dead_elapsed += Time.deltaTime;
	if(this.Dead_elapsed > this.Dead_Delay)
	{
		Dead_elapsed = 0;
		Application.LoadLevel("GameOver");
	}
}
}
```

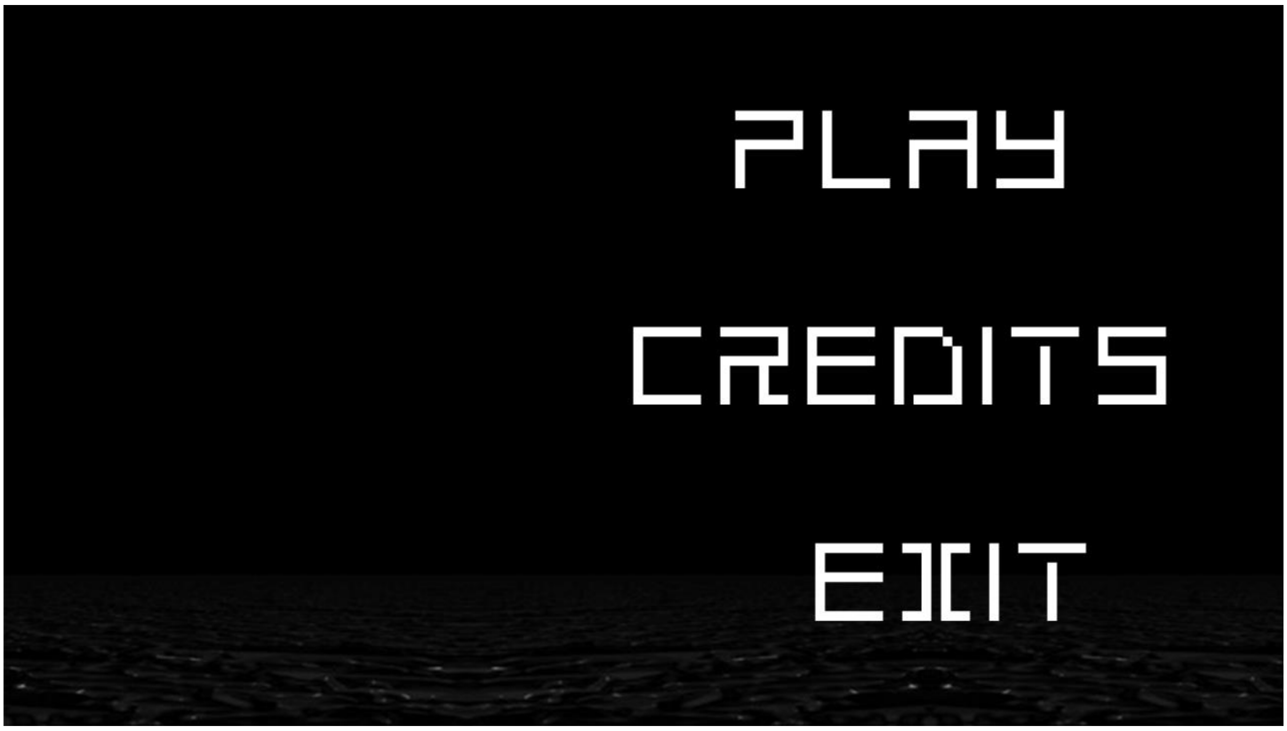
Platform_Manager.cs

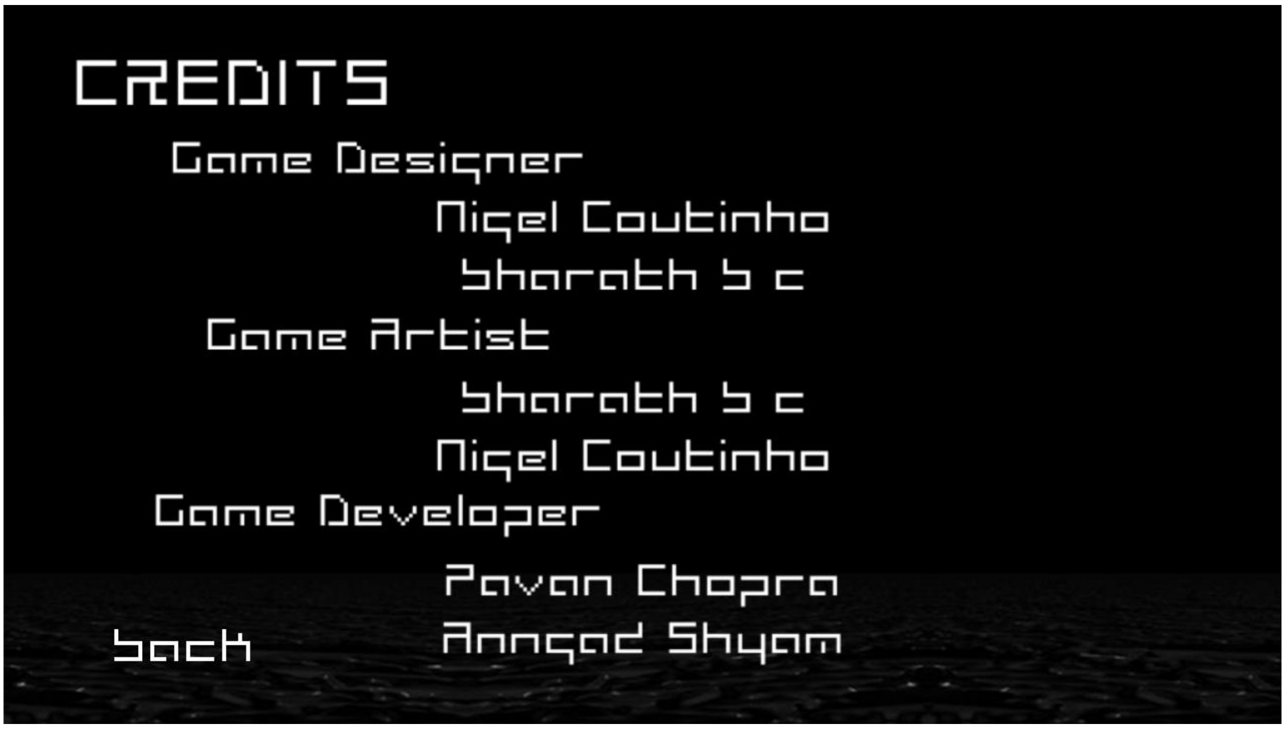
```
public class Platform_Manager : MonoBehaviour {
	public static bool move = true;
	public float speed;
	// Update is called once per frame
	void Update () {
		if(move)
		{
		// Translate the platform along Z=axis
		 transform.Translate(-Vector3.forward * Time.deltaTime * speed);
	    }
	}


}
```

Respawn.cs

```
public class Respawn : MonoBehaviour {

	public GameObject Platform_Prefab;
	public GameObject[] tempCar;
	public GameObject tempSet;
	public GameObject[] Set;

	// Use this for initialization
	void Start () {
	tempCar = GameObject.FindGameObjectsWithTag("Car");
	foreach (GameObject obj in tempCar)
	{
		if(obj.transform.parent == this.transform.parent)
		{
			Destroy(obj);
		}
	}
	int rand = Random.Range(0,2);
	tempSet = (GameObject)Instantiate(Set[rand],
this.transform.parent.transform.position,Quaternion.identity);
	tempSet.transform.parent = this.transform.parent.transform;
	}

	void OnTriggerEnter(Collider obj)
	{
		Debug.Log("Triggred!! " + obj.gameObject.name);
		if(obj.gameObject.tag == "Player")
		{
			Instantiate(Platform_Prefab,
transform.parent.FindChild("Spawn_Point").transform.position,Quaternion.identity);
			Debug.Log("Spawned!!!!");
		}
	}
}
```

SCREENSHOTS




CREDITS

Game Designer

Nigel Coutinho

Sharath B C

Game Artist

Sharath B C

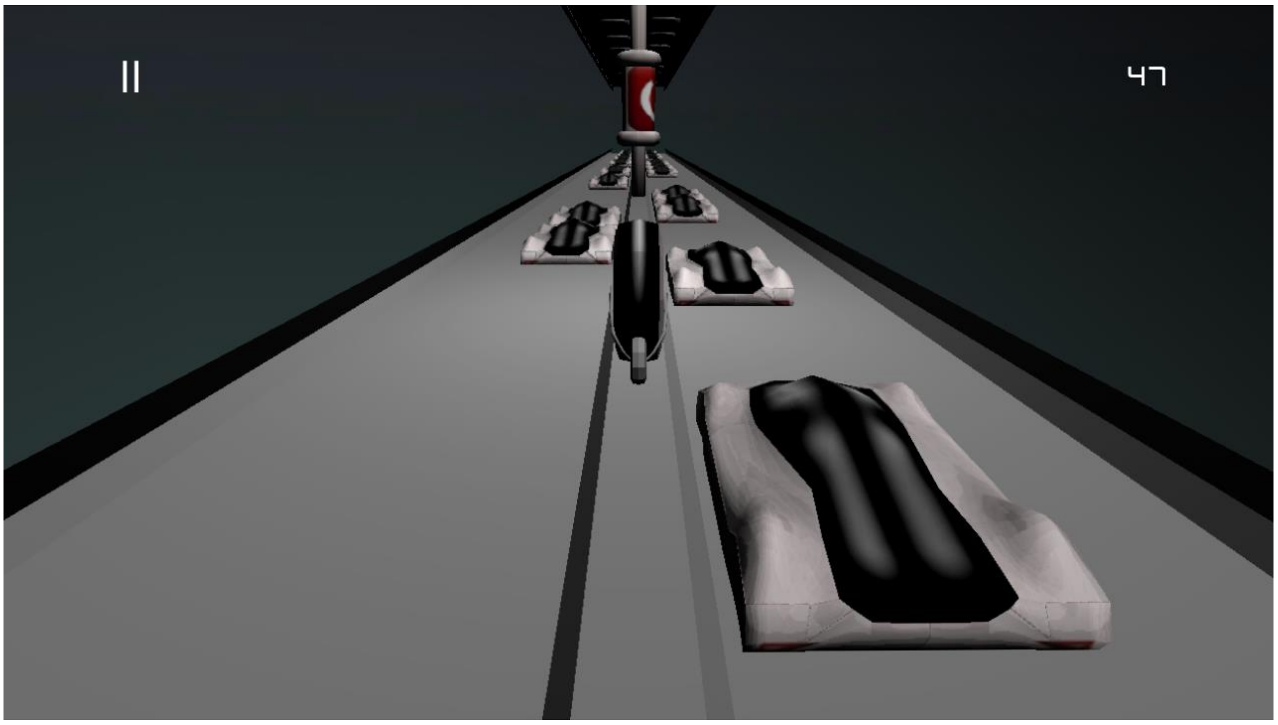
Nigel Coutinho

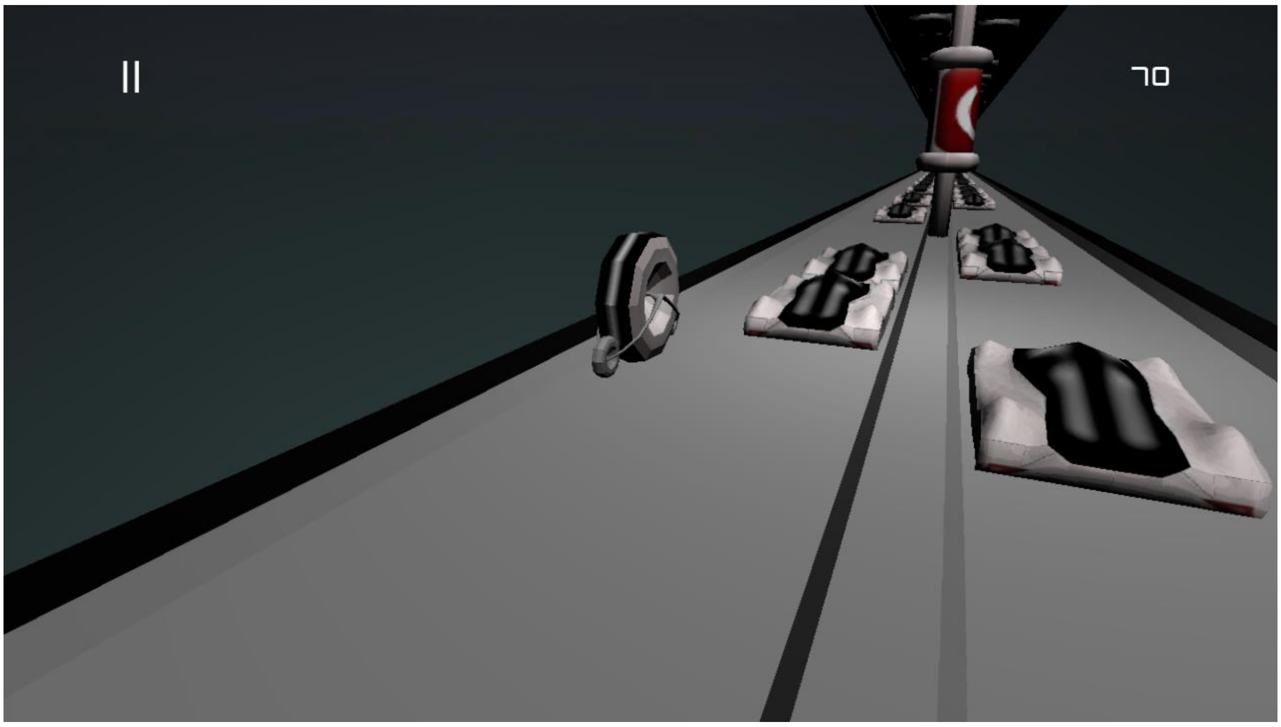
Game Developer

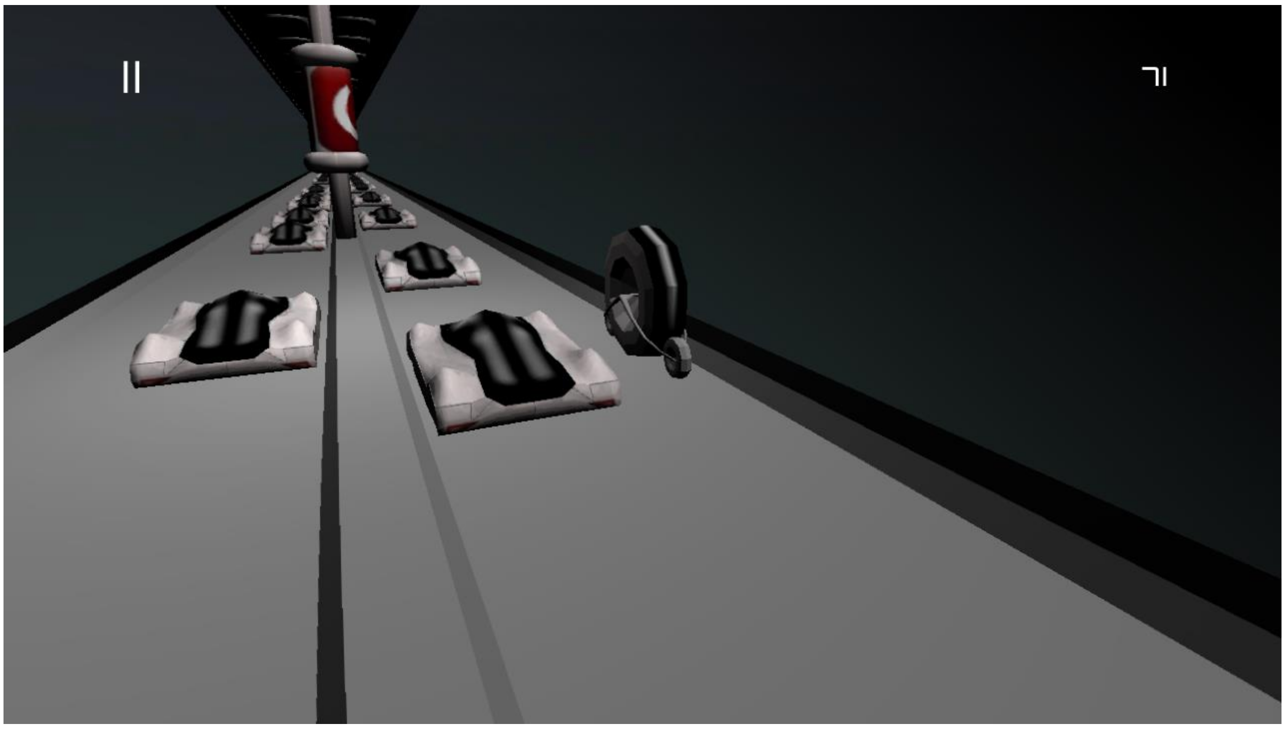
Pavan Chopra

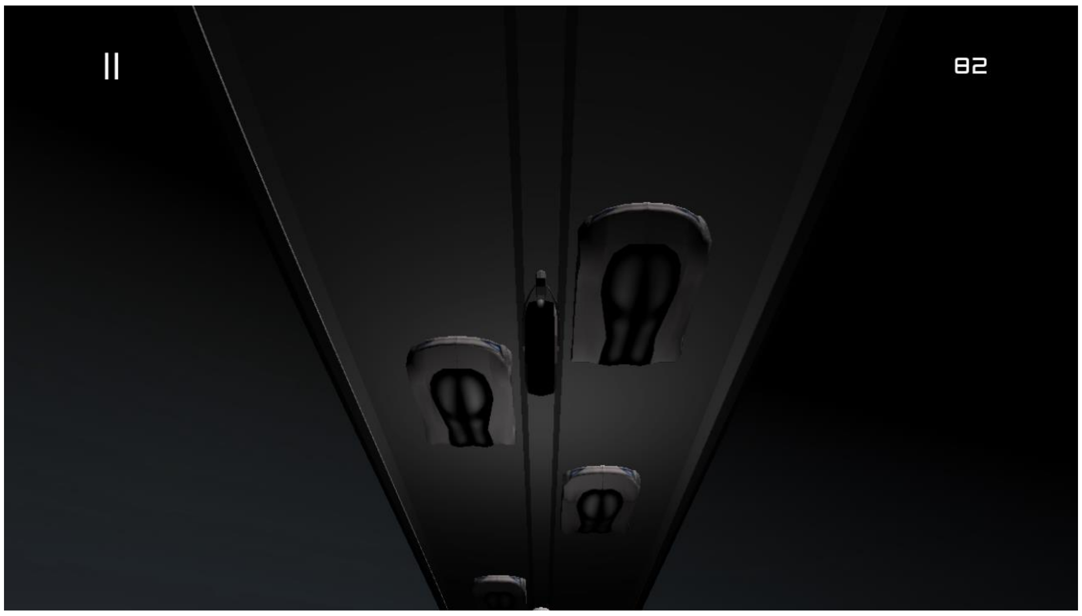
Anngad Shyam

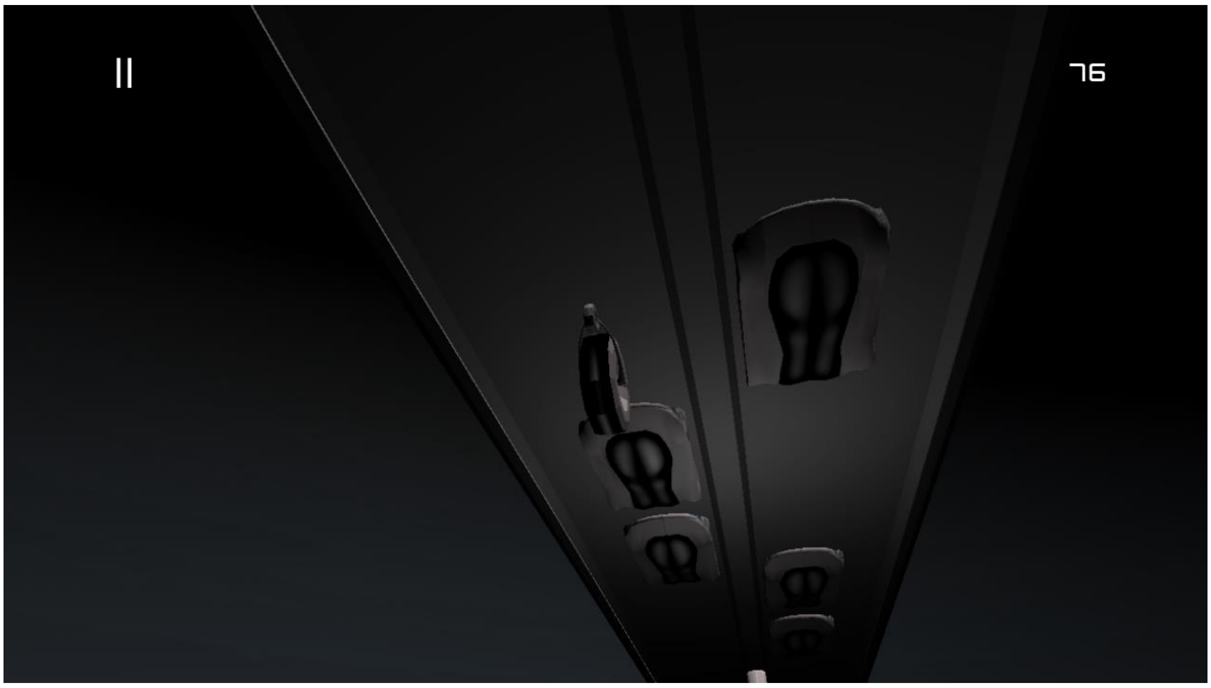
Back


||
47


||
70


II
71


II
82


II
76

Game Over
SCORE: 64
back