Computer Graphics and Animation

Report

KF6018

Module Tutuor: EdmondHo

Students: Visan Sebastian (w18016914)

Prahoveanu Oana-Diana (w18016671)

Neagu Alexandru-Adonis(w18017575)

Sebastian Visan (w18016914)

(Student 1 mission)

SkyBox

When choosing how to create the environment for the game the first step was creating a sky box which in our case is more of a sky sphere this was done by selecting a plain sky texture creating a sphere as the boundary for the environment and displaying the texture on the inside. Other considerations for the sky were a box texture on the inside, or a cube maped background, the sphere was used in order to make the sky look seamless as the other variants had clear edges.

Field and grass and ball

Both of these are simple planes with textures credits to @dgim-studio (field) and **AMOROMANIA** (grass). Worth mentioning is the code for a random hill generator that uses peril noise which is still present in the source code but has not been used since it did not create aesthetically pleasing terrain all the time and could some time result in cliping with the field, this also made the placing of objects quite difficult.

Lighting and shadows

In order to simulate direct sunlight a hemispheric light along with a directional light with rays coming from an angle as to simulate sunlight coming from the sun model in the sky. The directional light is also used to cast shadows as a hemispheric light unable to do so. The shadow camera map covers the whole field and as well as part of the sky box. The shadow map used is PCFshadows. and the resolution is 2000 by 2000 which should be lowered in case it has a noticeable performance impact.

Collision box

A collision box made from a plane and 2 cylinders is placed where the goal post is in order to help my colleague to stop the from going outside the net. The meshes cover the net area of the goal post and are transparent

Created objects .

Flags and flagpoles

These models are made using basic three geometry and the textures for the panels and flags were custom made by Diana Prahoveanu.

Other models imported using the GLTF loader. Credit for the model’s authors: **3DDomino** (Goal post), **StefanD** [(tribune)](https://sketchfab.com/3DDomino) **Sereno Kneringer** [(stylized tree)](https://sketchfab.com/3DDomino) **Dario.Aparicio.Valentin** (Cartoon Tree),**Hyungjung Kim** (low poly cloud), **Axt** (sun), [Casey Tumbers](https://poly.google.com/user/6q1z6TW2l0B) (trophy), Poly by Google(balloon)

Oana-Diana Prahoveanu(w18016671)

(Student 2 mission)

**Special effects:**

1.Flag

In order to create a realistic effect of the moving flags, the flag geometry vertices are used to add 3 waves. The variable multi is used to keep still the side of the flag where the flagpole is connected to the flag.

2.Billboard

In order to create the slideshow effect of the billboards, similar to the football stadium billboards, a variable is used to count how many frames each image should appear on the billboard plane.

3.Clouds

To create a realistic effect of the environment, the clouds are slowly moving on the x axis each of them with a different speed. Moreover, each cloud has a different path length in order to shift them.

4.Light

To create the effect of time passing, the light position is slowly moved on the z axis using the Math.sin formula.

5. Hot Air Balloon

In order to create a more festive and colourful environment, a moving hot air balloon is added to the scene. The Math.sin formula is used to create the floating effect on the y axis.

**Feedback to the user:**

6. Score

The first visual feedback that appear in scene is the score. In order to create it, a dom element is used. The statement “document.getElementById” is used to set the innerHTML of the div into the desired message. The score variable is increased with 50 points when the player catches the ball. In order to do so, collision variable is used to register if a collision happened between the player mesh and the ball, then count4 variable is used to take the first collision in order to increase the score.

7. Messages

A text-based feedback are the messages that appear under specific circumstances. For example, using the dom elements, messages are displayed in the fallowing circumstances: when the player catches the ball, when the player did not catch the ball, when the player leaves the goal post, when the player caught the maximum number of balls, and when the game is over because the player does not have any lives remained. A setTimout function is used for a part of the messages in order to hide them after a specific time.

8. Trophy

One of the graphic feedbacks to the user is when the player gets the maximum score, the game is over, and a trophy appear over the football field as a reward for the good skills of the player. In order to create the effect, the trophy object is moved on the y axis till it gets to the desired position.

9. Fireworks

When the player scores the maximum number of points a firework show is displayed. In order to create the firework effect, multiple particles are launched from random positions. When the launched particles get to a specific position the exploding effect is created. For each explosion, 50 particles are used to disperse on the x, y, z axis and they receive various colours.

10. Lives

Another graphic feedback to the user is the game lives. Similar to many games the user has 5 lives, each of them is represented by a spinning football. If the player does not catch the ball, one of the football disappears. When all five balls disappear, the player lost the game.

Neagu Alexandru Adonis (w18017575)

Functionality (Student 3 mission)

For this project, my responsibilities were to analyze and implement different kinds of movements on 3D objects. Since the main idea of the game is a football game, I chose to add a human-like 3D object which represents the goalkeeper. Also, I decided to create a crowd of supporters. The models and the animations that I used are fbx files which were downloaded from Mixamo. There are 10 different motions that I applied for the goalkeeper character in the game so that the player can have a more varied range of controls to defend the goal post. These animations allow the player to walk, run jump and dive after the ball. Furthermore, other 5 animations are implemented in the crowd like cheering and applauding.

Now that the models and animations were loaded, I created a few basic classes. BasicCharacterController class which is going to represent a single animated character in our little world. Will also make another class BasicCharacterControlerInput which will be responsible for listening for keyboard input and recording keys like up and down. Lastly, I made a FinateStateMachine and State class. The basic character is an instance as an instance of an input which will instantiate in the constructor and it has a finite state machine the character controller will load the FBX files associated with model and animations and then create a proxy class who will let the finite state machine communicate with the controller to change animations. The loading is all pretty simple we load the main model file then we load each of the animations and we store the animation clip and animation action from all of these net dictionary called this.\_animations. The keyboard input class is simple all this does is set a few document listeners up for the key-up and key-down events and record the relevant ones. We are mostly just interested in walking, running so will record a WASD for walking, 4,5,6,7,8,9 for defending the ball from num pad and the shift key for transitioning from walking to running. The input here will get directly passed to the finite state machine to be interpreted. If you were to press forward or backwards, he would be walking so we had this walk state and you transitioned to that from the idle state when the import was forward or backwards similarly when you are walking, and you stop pressing those keys we transition back into the idle state. The character states are walk, run, idle, walk\_back, run\_back, jump\_right, jump\_left, jump\_up\_right, jump\_up\_left, jump\_up and catch. The idea for this part of the project was inspired by a youtube video posted by SimonDev.

I also took care of ball animation. The ball will move from the centre of the football field towards the goalkeeper. It will get random directions by using the getRandomArbitrary function. When the collision is detected between the goalkeeper and the ball a function called resolve collision will make the ball to bounce in a different direction. The same principle applies to the goal post. When the collision is detected with the net of the goal post or the ball bounce of the ball will respawn back in the middle of the field and will come again towards the goal post. When the game ends the ball will stop the animation and will disappear.