3D CHARACTER ANIMATION

MANASA KONE
manasa.kone001@umb.edu
University of Massachusetts Boston

Figure 1: A scene upon loading 3D Character Animation

ABSTRACT

In this project we will take a look at creating a basic character controller using finite state machine. so that we can can instantiate characters in the little 3D world and you can move around. Also a realistic character is created in the blender and then rendered in WebGL. Three.js is used to create the scene.

KEYWORDS

WebGL, Visualization, Three.js, Blender, glTF

ACM Reference Format:
MANASA KONE. 2022. 3D CHARACTER ANIMATION. In *CS460: Computer Graphics at UMass Boston, Fall 2022.* Boston, MA, USA, 3 pages. https://CS460.org

CS460, Fall 2022, Boston, MA

ACM ISBN 1337.
https://CS460.org

1 INTRODUCTION

Animation is used on the TV, on phones, and all over the internet. I am more fascinated to create the animated model. So, now I got chance to create the animation character on my own. Specifically in my project, making our character stand around, walk, run and do a little dance.

W- walk forward
S - walk backward
shift + W - run forward
shift + s - run backward
A - rotate clockwise
D - rotate anticlockwise
space - to dance

2 RELATED WORK

https://threejs.org/examples/#webgl_animation_skinning_blending
Created 3D character animation using blender.

3 METHOD

- Firstly, in blender we work with the process of modelling where we create an 3D model of our own. Once creating the

model is done successfully it's' time to provide clothes to your 3D models.
- Next is the process of rigging and skinning. Its time to acquire a unique skeleton. A virtual skeleton is nothing but a rig. During rigging, all the body parts of the model are integrated as a whole body.In order to make the structure of 3D characters more realistic, our main goal is to make the model move. Must proceed with skinning after rigging, which involves integrating all of the 3D models with a rigged skeleton. This phase serves to handle the rig so that the 3D model may be changed quickly.
- Finally Animation, it is one of the most critical and time-consuming tasks that will add life to my 3D model.
- Now my model is ready to be exported out. select every part of the character and make sure every part is selected and now pre file exported and choose fbx while saving.
- we have got our animation data. lets create basic classes. They are:
- 1st class: BasicCharactercontroller class which is going to represent a single animated character.This has instance of an input which will instantiate as this.input in the constructor and also has an finite state machine.the character controller will load fbx files of a model and animations and then creates a proxy class that will let finite state machine communicate with the controller to change animations. Then we load each of the animations and we store the animation clip and animation actions from all of these in a dictionary called .animations.we use lodemodels function where loading is all pretty simple by loading the main model file.
- 2nd class: basiccharactercontrollerinput which is responsible for listening for keyboard input and recording keys like up and down.I mostly interested in walking, running and dancing. So, I will record a, w,s, d and space key for doing dance and shift key from walking to running. The input here will get directly passed to the finite state machine.
- 3rd class: Finitestatemachine class these are really simple way of modelling the possible states and transitions for a character.There are few public functions. They are: 1) Set state function which first signals the old state that exiting and sets the current state and then notifies the new state that is active. 2) Update function that gets called every frame and passes the frame time and input to the currently active state.
- Now we make character fsm class which will be just a child of general finite state machine and I just do that so I can add the character specific states. The add state calls were passing the name of the state and then the class that will get instantiated to represent that state. For instance when we switch to idle the finite state machine will instantiate an idle state instance.
- State Transitions: Beginning from the idle state- when we enter the idle state. we want to set the characters animation to idle. Enter(prevstate) function is responsible for playing the idle animation. If there was a previous state i.e if character is walking before all this code here does is tries to cross fade them old fades out and new one fades in. update function can transition to either dance or walk.
- ———Three.js and WebGL Implementation———-
- Using effective camera angles and approaches for making 3D character animation films is important, in as well as taking lightning into account.One of the final phases in the pipeline for 3D animation is rendering. This stage involves translating all the data into the file so that every shot may be combined into a single frame.It's crucial to consider all the factors, including camera positioning, whenever you create a final render of a scene. Compositing must be done after rendering is complete.
- The approach outlined above is how I approached my 3D CHARACTER ANIMATION.

3.1 Implementation

Below is the implementation of the loadModels()

```
_LoadModels() {
const loader = new FBXLoader();
loader.setPath('./resources/zombie/');
loader.load('manasa.fbx', (fbx) => {
  fbx.scale.setScalar(0.1);
  fbx.traverse(c => {
    c.castShadow = true;
  });

  this._target = fbx;
  this._params.scene.add(this._target);

this._mixer = new THREE.AnimationMixer(this._target);

  this._manager = new THREE.LoadingManager();
  this._manager.onLoad = () => {
    this._stateMachine.SetState('idle');
  };

  const _OnLoad = (animName, anim) => {
    const clip = anim.animations[0];
    const action = this._mixer.clipAction(clip);

    this._animations[animName] = {
      clip: clip,
      action: action,
    };
  };

  const loader = new FBXLoader(this._manager);
  loader.setPath('./resources/zombie/');
loader.load('walk.fbx', (a) => { _OnLoad('walk', a); });
loader.load('run.fbx', (a) => { _OnLoad('run', a); });
loader.load('idle.fbx', (a) => { _OnLoad('idle', a); });
loader.load('dance.fbx', (a) => { _OnLoad('dance', a); });
});
}
```

3.2 Milestones

3.2.1 Milestone 1. Selected the 3D character animation from the final project list given.

3.2.2 Milestone 2. Devoloped a 3D character which includes the blender as the first step where we create 3D model, texturing, rigging and animations(walk,run,idle,dance).

3.2.3 Milestone 3. WebGl and three.js is used to design scene by adding camera, rendered and lights. For loading .glb file I used THREE.GLTFLoader() and THREE.TextureLoader() but with different approach for .glb file.

3.3 Challenges

- Challenge 1: In the process of creating a 3D human model with the creation of unique skeleton during the rigging process and giving life to such unique skeleton during the animation process, working on Blender as a beginner was challenging and time-consuming.
- Challenge 2: Blender and WebGL integration extremely challenging.

4 RESULTS

5 CONCLUSIONS

In conclusion, I have successful in creating the 3D character animation. Working with the bender is incredibly helpful and informative during this project, and the tutorial guided me through a few key features that I was glad to implement. Through this project, while three.js and WebGL has integrated, I gained knowledge of the gltf file format, loaders, and built-in animations in three.js. In this project my project's main goal is to make my character walking, running, and even doing a little dancing according to the keys we click, walking. May be in future I will try to develop the features of my project to make it highly productive and exciting.

REFERENCES

https://threejs.org/examples/#webgl_animation_skinning_blending https://tympanus.net/codrops/category/tutorials/ https://threejs.org/examples/#webgl_animation_keyframes