Sapienza University of Rome

Interactive Computer Graphics

Sino: Haymaking challenge

Author:
Horpynchenko, Dmytro

Student Number:
1807584

Contents

1 Intro

Final project is a browser game with elements of 3D graphics.

1.1 Concept

The world of the game is a field with hay blocks and obstacles on it. A user is a driver of a vehicle that can move on the field in any direction and collect a hay. In case of collision with the obstacle, vehicle loose all its speed or accelerates in opposite direction in case of high speed. The aim of the game is to collect all hay blocks in defined time period. Amount of time, hay blocks and obstacles depends on the level of difficulty chosen by user.
The vehicle has 2 controls available:

- Front wheels steering (left and right) controlled with left and right arrow keys. After release of the keys vehicle's wheels return to their default positions of 0 deg.
- Accelerating (forward and backward) using up and down arrow keys accordingly. Release of the keys stop acceleration and leads to deceleration of the vehicle because of the wheel's friction.

Collecting action is done by colliding with hay block that lead to increase of the score and removing hay item from the field.

1.2 Name

The name "Sino" comes fron Ukrainian word "сіно" - hay [1].

2 Implementation

2.1 Overview

2.1.1 World

The world of the game is represented as a simulation of real world with infinite grass field and a blue sky around.

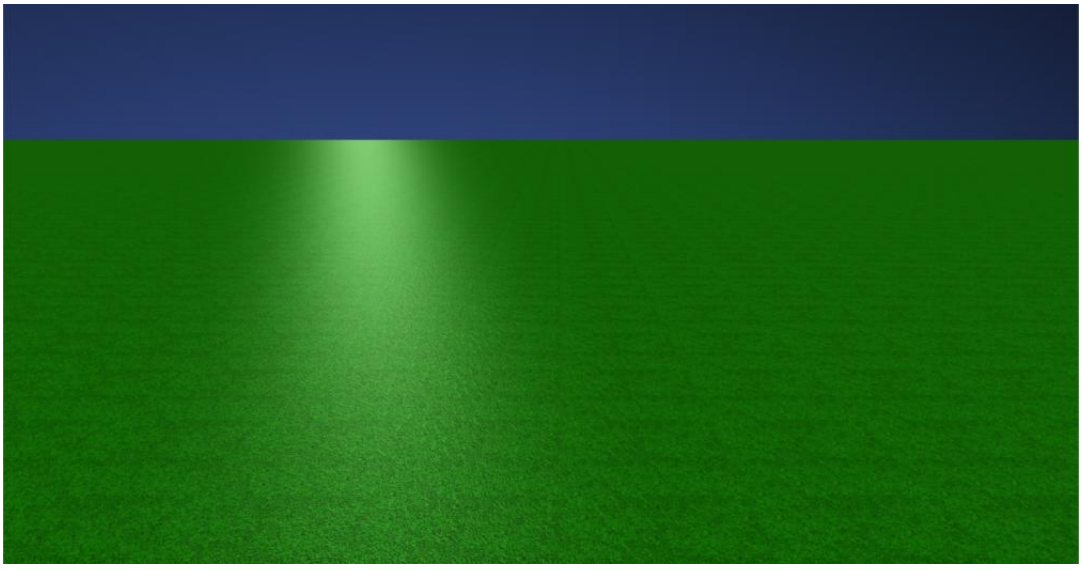

Figure 1: Game world

2.1.2 Hays and obstacles

Hays are represented as parallelogram of constant size 1x1x2 with texture of hay (Figure 2).

Obstacles are represented as parallelograms with various sizes. An example of obstacle of size 1x1x4 is showed on Figure 3.

2.1.3 Vehicle

Vehicle model is a model found on the online marketplace TurboSquid (`www.turbosquid.com`). Model represented as tractor vehicle (Figure 4). For performing an animation

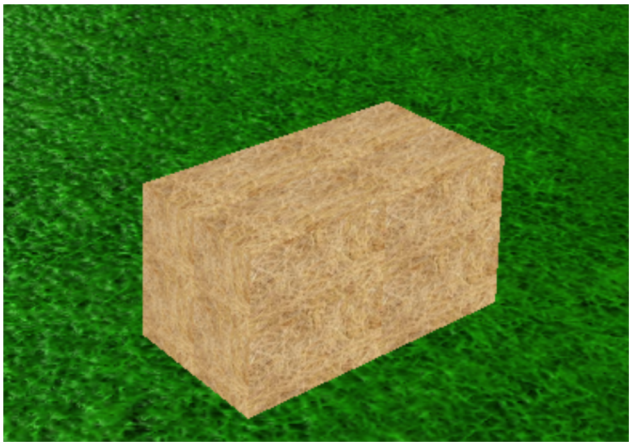

Figure 2: Hay item

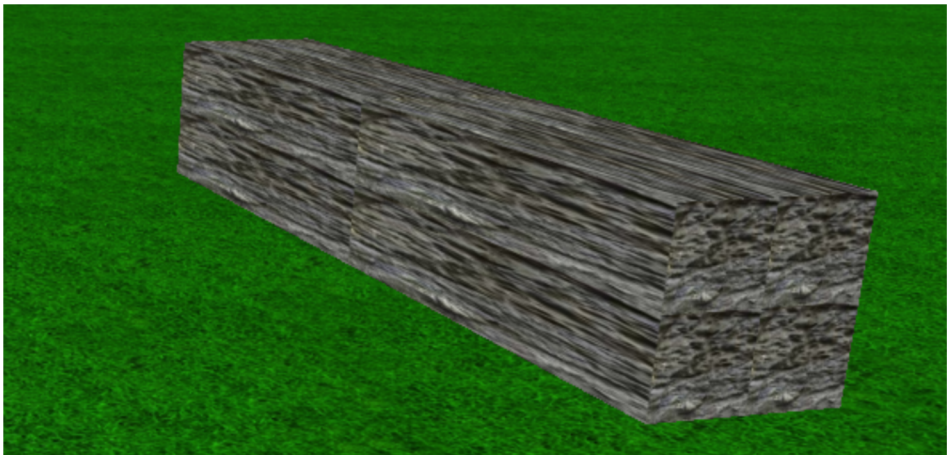

Figure 3: Obstacle item

3D model was separated using Blender software into body, front and rear wheels. Every part of the model is exported through Waveform .obj file format for later using during run time.

2.2 Libraries and Frameworks

All modern browsers has support of WebGL - a JavaScript API for rendering interactive 2D and 3D graphics with GPU acceleration[4]. But usage of WebGL is quite complicated, requiring writing a lot of boilerplate code, helper functions and hardly maintained code. Thus, in this project THREE.js (`https://threejs.org`) open source library was used that offers high-level object-oriented JavaScript API.

Figure 4: Tractor model in Blender

Physics emulation libraries like Physi.js or ammo.js weren't used since they are not necessary for game and make big impact on overall performance.
For debug purposes and performance tracking Stats library (`https://github.com/mrdoob/stats.js`) is used.

2.3 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications[3]. Project is represented as a single page web application with *index.html* as main entrance point and consists of two parts:

- Head, which includes data, like name of the web page, author and imports CSS and JavaScript files.
- Body that describes structural hierarchy of UI elements on the web page.

2.4 CSS

CSS (Cascading Style Sheets) is a technology that simplify description and declaration of so called "styles" or parameters of elements of HTML page[2]. "Sino" project contains *styles.css* file under css folder that describes parameters of elements of html page. This includes position inside parent, text color and size, paddings and margins, scales etc.
As a font ArcadeClassic was used.

```
@font-face {
    font-family: 'ArcadeClassic';
    src: url('../font/ARCADECLASSIC.woff');
}
```

Listing 1: Declaration of custom font

2.5 JavaScript

All algorithm parts are written in JavaScript and consist of few separate js files for easy maintenance.

2.5.1 game.js

Entrance point of the game's JavaScript code. Contains game UI initialization, game logic and communication between all other elements of the code. Includes following functions:

- *init* - handles creation and setup all components of the program after start.
- *initScene* - called during init function execution. Initialize and configure main components of THREE.js library renderer and camera.
- *initUI* - called during init function execution. Initialize sreen UI elements like labels, buttons and their callbacks, Stat library.
- *addLight* - alled during init function execution. Initialize, configure and add to scene 3 light source: ambient light using lib THREE.js class *AmbientLight* for creating uniform light all along the scene and 2 directional lights from behind and front using class *DirectionalLight*.

```
function addLight() {
    //    set 3 point lighting
    // add a ambient light
    var light = new THREE.AmbientLight(0x020202);
    scene.add(light);
    // add a light in front
    light = new THREE.DirectionalLight('white', 1);
    light.position.set(0.5, 0.5, 2);
    scene.add(light);
    // add a light behind
    light = new THREE.DirectionalLight('white', 0.75);
    light.position.set(-0.5, 2.5, -2);
    scene.add(light);
}
```

Listing 2: Adding light sources

- *initRendering* - called during init function execution. Describe sequence of operations done for rendering each frame. Call a system function *requestAnimationFrame* to subscribe for next frame update.
- *onWindowResize* - called during init function execution. Called each time size of the browser window is changed. Adjust renderer and camera output parameters according to new window dimensions.

```
function onWindowResize() {
    camera.aspect = window.innerWidth / window.innerHeight;
    camera.updateProjectionMatrix();

    renderer.setSize(window.innerWidth, window.innerHeight);
}
```

Listing 3: onWindowResize function

- *update* - called on every frame update call.

2.5.2 core.js

Contains main components of the code

2.5.3 items.js

Maintains initialization of graphical components of the game

2.5.4 utils.js

Consist of utility functions that are not depend of the logic and structure of the game.

2.6 Distribution

Final source code of the project is public accessible as git repository on the GitHub web service by `https://github.com/MarcoSchaerfCourses/sino/` link. It could be run online by link `https://marcoschaerfcourses.github.io/sino/` since repository was deployed using GitHub Web Pages service.

3 Known Issues

Testing of final product showed following issues that require additional time to investigation:

- Performance issues
- Texture import
- Bounding Box size increasing depending on rotation of object
- Adaptation for small screen sizes

4 Possible Improvements

4.1 Performance

4.2 Interface

References

[1] Wikipedia, The Free Encyclopedia. , 2019. [Online; accessed 12-January-2019].

[2] Wikipedia, The Free Encyclopedia. CSS, 2019. [Online; accessed 12-January-2019].

[3] Wikipedia, The Free Encyclopedia. HTML, 2019. [Online; accessed 12-January-2019].

[4] Wikipedia, The Free Encyclopedia. WebGL, 2019. [Online; accessed 12-January-2019].