Farm 4 money

12.06.18

Dominik Hanko
01625726

Yanik Thurner
01625761

Features

Fields

In the game there are fields on the map where seeds can be planted. Over time the plant grows and when it reaches it final stage it can be harvested. The final stage is shown by the particle system on the corresponding plant. For now the player grow a single type of plant and has infinite seeds. The player can plant seeds by pressing “F”.

Fishing

By going near the water and pressing f a small mini game starts where the player has to press when the right picture shows up to catch a fish. It can take a while if you are unlucky as fish bite by random. The caught fish can then be sold for additional money.

Inventory

The player can see the inventory at the bottom of the screen. The selected item has an red overlay and can be changed with the mouse wheel. The money is displayed on the right side. It can be upgraded to upto 10 slots by buying the upgrade items at the shop.

Selling and buying Items

Items can be sold and bought at the respective box that is placed on the map. Left besides the lantern is the shopbox and on the right is the shipping box where items can be sold. The player sells the first item in the inventory by getting near the box and pressing “F”. To buy items the according number key has to be pressed.

Jumping/Player gravity

The player is under gravity influence under all times, therefore the player can fall down from objects. Additionally the player can jump nad overcome the gravity for a certain amount of time.

Requirements

1. Gameplay

The Gameplay consists of planting seeds and harvesting plants. To abridge the waiting time the play can catch fish while waiting for the plants to grow. The yield can then be sold at the respective chest (the right one in this build) and other seeds can be bought at the other box. The goal for the player is to buy the graphics card and finish the game.

2. Compulsory effects

A. Light Mapping

The ground-texture uses light mapping rendered in Blender. The fields also have a light map on them but only one texture for every field.

B. Heads-UP Display

The HUD displays the inventory and available cash.

C. Physics Engine

PhysX is used for all collision detection, character movement and interactions which are done with trigger shapes.

3. Complex 3D Models

All models with their texture or color data are imported by assimp, so complex models can be used.

4. Hierarchical Animation

We tried implementing Skeletal animation, but there were some problems and id didn’t quite work right, so we had to choose an alternative. The player head can be moved by pressing alt and turning the camera.

5. Frustum Culling

Frustum Culling is implemented and can be switched on and of by the F8 key. The distance can be set via the settings.ini file

6. Controls

WASD - player movement

Space - Jump

E - show inventory in cmd (legacy)

F - interact (plant seeds, harvest, fish, interact with trader)

P - cheat money

L ALT - hierarchical head transform

Mouse - camera movement

ESC - end game

Mouse wheel - browse inventory

7. Debug Options

F2 - enable/disable Framerate indicator

F3 - enable/disable Wire Frame

F4 - enable/disable HUD

F8 - enable/disable Frustum culling

Effects

1. Cel Shading + Edge Detection (1.5P) - Hanko Dominik

First the light intensity is calculated after the phong shading model. Then the Intensity is multiplied by the amount of shading levels. This value is rounded down and divided by the amount of shading levels. As a result of this calculation, the lighting intensity is now divided into the given levels.

The outlines are added in post processing. The shader returns not only the color but also an image of the normals. An edge detection algorithm is then used on this image and the found edges are added to the color image.

2. Particle System (0.5P) - Hanko Dominik

In the particle system, many instances of one object are created. Every instance of these objects has the same vertex data but a different position and move_direction. Therefore 2 buffers are created one static_draw buffer for the vertex data and one stream_draw buffer which is updated every frame.

3. Water Mesh (1P) - Thurner Yanik

Water Mesh is implemented as demonstrated in the Repetitorium. We use a plane as a model and change the y positions in a shader according to a simple sine function. Then the normals are corrected so that the shading works as intended.

4. Video Texture (1P) - Thurner Yanik

We loaded the video file with opencv (because after trying a long time we gave up on ffmpeg), gather each frame and load it as a texture. We then have an array with each texture handel and a pointer with the current frame, which we draw as long as we let it point to the next one in our update method to keep it frame independent.

Illumination

The whole scene is lit by ambient directional light. Additional there are 5 lamps placed throughout the scene which work as point lights.

Used Tools

- Blender

Additional libraries

Nvidia PhysX 3.4

Used for physics (collision,..) and trigger shapes.

https://developer.nvidia.com/physx-source-github

assimp 4.0.1

Used to import .blend files

http://www.assimp.org/

stb_image

Used to read images when reading model textures and images for the HUD.

https://github.com/nothings/stb

freetype

Used to generate bitmaps from fonts for later display.

https://www.freetype.org/

opencv

Used to read videos for the video texture effect.

https://opencv.org/

Used tutorials/Code examples

I. ECG Framework

Framework was provided by the ECG team.

II. learnopengl.com

Used the Model tutorial for importing models with assimp.
https://learnopengl.com/Model-Loading/Model

Copyright

Character model

License: CC-BY

https://www.blendswap.com/blends/view/83087

fence model

License: CC-BY-NC

https://www.blendswap.com/blends/view/76557

Grass Texture

Hand Painted Grass Texture from Unity-Store:
https://assetstore.unity.com/packages/2d/textures-materials/floors/hand-painted-grass-texture-78552

Bäume und Steine

License: CC-BY

https://www.blendswap.com/blends/view/83490