Farmer Run - COSC 3P98 Project

Adam Balint
5141619
2016/01/11

The Game:

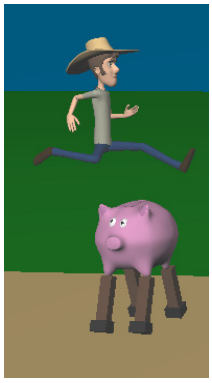
Controls:

Press and hold the up arrow to jump, the down arrow to slide and press the "R" key to restart when you lost the game. Pressing escape will quit the game. The score and any level ups are displayed in the console

The Enemies:

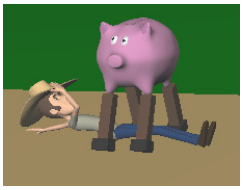
Sheep:

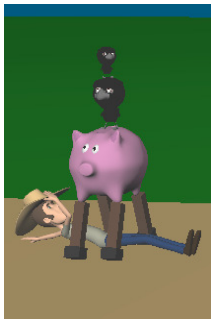
These sheep are just like any other normal sheep. Jump to avoid them!



Pigs:

These crazy pigs on stilts think they are smart until they see you running by them. You can see the look of fear in their eyes hoping that you don't notice them. You can jump over or slide under these troublemakers to avoid them!





Pigs & Birds:

These pigs with birds are slightly more intelligent than their counterparts above. They have called their friends, the birds, to give them some camouflage. With their fear struck face, the only way to avoid these mischievous creatures is to slide under them!

Note: The game takes a while to start initially because it needs to load all of the models. Please be patient.

The Implementation:

Classes:

Source:

The main game code. This file contains the user interaction and game logic.

ModelLoader:

This class is responsible for loading all of the models.

Model:

This class holds all of the vertices, material and lighting information.

Material:

This class holds all the information related to the lighting.

Player:

This class holds all of the information and logic for the player. This includes animation and checking is the player can perform certain actions.

Entity:

This class holds all the information in relation to entities.

Model Loading:

Model loading is done using .obj and .mtl files exported from blender. These files need to be exported with only the "Apply Modifiers", "Write Normals", "Write Materials" and "Triangulate Faces" options selected. Once exported, the ModelLoader class will parse through the file starting with the obj file and generate all of the required vertices, normals and faces. It will also keep track of which materials will be used for each face. All of these are added to a Model object. Since, models can be made up of multiple objects, each subset of faces using the same material are grouped together under a "Face set". Next, the class moves on to reading the material file. A material is then generated. The name of the material is kept track of, as after the material is completed, the material is added to the model and placed into a table using the name of the material as the key. When the model is drawn, it combines all of the above information to draw the model.

The Keywords:

Object File(.obj)

mtllib - this identifier states that the following value will the name of the material file to use with this model.

usemtl - this identifier specifies which material to use in the material file, as one material file may contain more than one material.

v - this identifier states that the following 3 values will be x, y and z co-ordinates of a vertex.

vn - this identifier specifies that the following 3 values will be the nx, ny and nz values of the normal.

vt - this identifier specifies that the following 3 values will the u, v, and w co-ordinates of the texture

f - this identifier specifies the following values construct a face using the indices of the vertex, vertex textures and vertex normals in that order. The vertex textures area may be empty if no texture is used. Each of the values a separated by a forward slash ('/')

Material File(.mtl)

newmtl - the identifier that states that this is a new material. This identifier is followed by a material name.

Ns - the identifier that indicates that the shininess value will follow. This identifier is followed by one value between 0 and 1000. This value is divided by 1000 in order to implement into the OpenGL range of 0 - 1

Ka - the identifier that indicates that the ambient lighting values will follow. This identifier is followed by 3 floats containing the ambient RGB values.

Kd - the identifier that indicates that the diffuse lighting values will follow. This identifier is followed by 3 floats containing the diffuse RGB values.

Ks - the identifier that indicates that the specular lighting values will follow. This identifier is followed by 3 floats containing the specular RGB values.

d - the identifier that indicates that the dissolve(alpha) values will follow. This identifier is followed by one float. This value is applied to the alpha channel of the diffuse, specular and ambient lighting properties.

An example can be seen on the print out attached to the end of this document.

The Animation:

The animation just consists of multiple models that have been animated. These models are displayed at for two game frames before moving onto the next frame. This provides a slower animation which suits the environment better. The models themselves are not stored in the player class; the player class just keeps track of the location of each type of model, with the models for the animation just being a displacement from the first pose. The player keeps track of this displacement in order to provide which model to use.

The Game:

The game is responsible for drawing all of the UI, and checking for user input. The game is also responsible for deleting and creating enemies, keeping track of the score, and checking for collisions.

Models:

Person:

Made by: VMComix
Download: http://www.blendswap.com/blends/view/24092
Changes:
- Added hat (Download: http://www.blendswap.com/blends/view/76228)
- Changed clothes colour (including the hat)
- Animated

Sheep:

Made by: hamburger891
Download: http://www.blendswap.com/blends/view/77070

Pig:

Download: http://www.turbosquid.com/3d-models/piggy-bank-obj-free/567429
Changes:
- Changed the colour
- Filled in the coin slot
- Added eyes
- Added stilts

Bird:

Made by: WIRLOW
Download: http://www.blendswap.com/blends/view/21221
Changes:
- Reduced Polygon count and combined with pig

Cloud:

Made by: jaimin9475
Download: http://www.blendswap.com/blends/view/72484

Future Improvements

Some future improvements include:

- improve animation
- add in animation for sliding and jumping
- add texture functionality to ModelLoader
- add edge mapping functionality to ModelLoader
- improve collision detection

References:

http://www.blendswap.com/
http://www.turbosquid.com/index.cfm
http://paulbourke.net/dataformats/mtl/
https://en.wikipedia.org/wiki/Wavefront_.obj_file
http://www.andrewnoske.com/wiki/OBJ_file_format