PA7: Solar System

By Ronn Quijada and Sean Stevens

User Manual

To compile:

Create a directory in PA7 called “build”

Go to the “build” directory

In the command terminal, write “cmake ..”

Then “make”

Drag the assets folder into the “build” directory.

If it was successful, enter “./Tutorial” to run the solar system program

If the program segfaults, try using the “SUPER low-res assets” pack. You have to rename the folder to “assets” when in the “build” directory.

Note: This program needs GLEW, GLM, SDL2, and ImageMagik.

Controls:

Esc – quit program

Q – speed up simulation.

E – slow down simulation.

Space – Pause simulation.

Right Mouse Button Down – Pause rotation

Left Mouse Button Down – Play rotation

Documentation

Camera.h

Camera for what the user is looking at

Initialize (int w, int h) Creates camera with given width (w) and height (h) parameters. Initializes the view and projection matracies

GetProjection () Returns the projection matrix

GetVIew () Returns the view matrix

SetParent (Object*) Sets object for camera to focus on

SetWorld (Object*) Sets object for camera to look at if no parent is selected

Update (unsigned int dt) Handles event for keyboard interaction

engine.h

Class for the game engine

Engine (string name, int width, int height) Creates engine, parameters set window to specified dimensions.

Engine (string name) Parameterized constructor that sets window to default size

~Engine () Deletes engine instance and instances of engine, event, and graphics

Initialize () Initializes the graphics, window, and event handler. Returns true if successful.

Run () Updates the window, graphics, and camera

Keyboard (eventType) Event handler for keyboard interaction

getDT () returns time between this frame and the previous frame

GetCurrentTimeMillis () returns the system time in milliseconds

event.h

Class for handling events

update () – Function to process eventQueue

event () – construct event handler class

pushEvent (Uint32, Uint8, SDL_Keycode, Sint32, Sint32) Adds event to queue

graphics.h

Class for rendering objects

Initialize (int w, int h) Initializes graphics instance with parameters for the dimensions

InitShader (Shader, string, string) Initializes a selected shader

generateFrameBuffer (GLuint, GLuint, int, int) Generates frame buffer with window size and target parameters.

Update (unsigned int) Updates objects and camera

Render () Renders the image and sends it to the string

addRenderTarget (Shader, GLuint) Enables the shader for a given texture

RenderList (vector<Object*>) Renders a vector of objects

TreeRender (Object* object) Selects a shader used to render the object

ErrorString (GLenum error) Returns a string that represents an error if there was one

graphics_headers.h

Class for openGL and other libraries

loader.h

Class for loading objects and textures

loadObject (string, obj&) Loads an object at specified file location. Returns true if the object was loaded successfully.

loadShader (string, string&) loads a shader from a file. If file was valid, function returns true and the shader will be passed back by reference.

loadTexture (string, Texture&) loads a texture from a file. Returns true if the location was valid and the texture is passed back by reference.

Everything else has been deprecated because assimp handles file loading

moon.h

Derives from object class. Special object that orbits a planet.

Moon () Default constructor. Also has parameterized constructors.

Moon (istream&, Object*) Constructor that takes in an istream so the object may be loaded from a config file. Object* is the parent to orbit.

Update (unsigned int) Moves the object around the parent based on given speed and orbit parameters.

setParent (Object* par) Sets parent to rotate around

setSize (flaot) sets size of moon

ostream& operator<< (ostream&, const Moon&) Operator overload to save moon to a file.

obj.h

Class to store data from obj file

addVert (Vertex) Adds a vertex to vertices list.

addVert (Vertex, int) Adds vertex at index

addIndice (unsigned int) Add index to indices list

addRaw (glm::vec3 raw) Adds raw vector to vertices

vector<Vertex> getVerts () returns a vector of vertices

vector <unsigned int> getIndices () returns a vector of indices

vector <glm::vec3> getRawVerts () returns a vector of raw vertices

object.h

Base object class that stores data texture and model data for objects

Init () initializes rott directory

setVisual (string, string, string) Loads model, albedo texture, and normal texture for object

loadNewModel (string) loads a model from a given path. Returns true if successful.

loadNewTexture (string) loads a texture from a given path. Returns true if successful.

loadNewTexture (string, int) loads a texture from a given path. Loads texture into index. Returns true if successful.

loadNewNormal (string) loads a normal map from a given path. Returns true if successful.

bindTex (GLuint&, GLenum) binds given texture.

setTex (Texture) sets parameters of a given texture.

Update (unsigned int) virtual function for updating model

setMultiplier (float) sets multiplier used for speed step

keyboard (eventType) virtual function for derived objects to handle events

GetModel () returns model matrix

Render () renders the object

getChildren () returns vector of children

isaPlanet () returns value of isPlanet variable

isaGasGiant () returns value of isGasGiant variable

setGasGiant (bool) sets value of isGasGiant variable

getSize () returns size of object

isEarth () returns value of earth variable

setEarth (bool) sets value of earth variable

isRing () returns value of isaRing variable

isaSkyBox () returns value of isSkyBox variable

setRing (bool) sets value of isaRing variable

setSkyBox (bool) sets value of setSkyBox value

getHorizon () returns color of planet horizon

getAtmosphere () returns color of atmosphere

setHorizon (glm::vec3) sets color of horizon

setAtmosphere (glm::vec3) sets color of atmosphere

planet.h

Planet () default planet constructor

Planet (istream&) constructor that passes an istream to load from file

Planet (float, float, float, float) parameterized constructor to set rotation, orbit, distance, and size

Plant (float, float, float, float, float, float) parameterized constructor to set rotation, orbit, distance, size, tilt, and offset

Update (unsigned int) Updates planet based on given parameters

setSize (float) sets size of planet

operator<< (ostream&, const Planet&) operator overload to save planet to a file

shader.h

Class for loading and initializing shader programs

init () Initalizes root directory

Initialize () creates a shader program. Returns true if successful.

AddShader (GLenum) Adds a vertex or fragment shader. Returns true if successful.

AddShader (GLenum, string) Adds a specified shader from a file. Returns true if successful.

Finalize () Link and validate loaded shaders.

Enable () Enables shader program

GetUniformLoacation (const char*) returns the location of a uniform value.

getShader () returns a shader program.

solarSystem.h

Derives from object class. Creates tree for other planets.

SolarSystem () default constructor

SolarSystem (float, float) Sets size and rotation speed of sun.

SaveSolSystem (string) Saves solar system to a file path.

LoadSolSystem (string) loads a solar system from a file.

LoadSolSystem (istream&) loads a solar system given an istream&

Update (unsigned int) Handles keyboard events and then updates solar system and children.

setSize (float) Sets size of solar system.

Keyboard (eventType) handles keyboard event.

ostream& operator<< (ostream&, const SolarSystem&) Outputs solar system to a file

istream& operator>> (istream&, SolarSystem&) Reads solar system from a file.

window.h

Initialize (const string, int*, int*) Creates a window with specified name and dimensions

Swap () Swaps the frame buffer.