GDPHYSX
Physics Engine Phase 1
Documentation

XX22
Bumanglag, Nikos
Domingo, Valentin

Public Functions:

float randomFloat(float min, float max);

- A function that takes a range of values and returns a float number within the range provided.

void generateRandomColor(GLfloat* color);

- A function that generates random value for color (r,g,b).

P6::MyVector generateRandomForce()

- A function that generates random numbers(x, y, z) for force to an object and returns it.

void Key_CallBack(GLFWwindow* window, int key, int scancode, int action, int mods);

- A function that detects key press/release and moves the camera accordingly (W,A,S,D). Also detects presses for pause and play, and switching between ortho and perspective view.

Object Class:

- The object class holds data that an object will need (ex: shader, position, scale, …, color)

Functions:

- **Object(GLfloat* color, Shader* shader);**
  - The constructor of the object.
  - Sets the color and the shader of the object.
- **void draw();**
  - The draw function updates the transformation matrix.
  - Where shader is being updated.

P6Particle Class

- The particle class that holds data that a particle needs (ex: mass, position, lifespan, …, damping).

Functions:

- **P6Particle();**
  - The default constructor of the class;
- **void AddForce(MyVector force);**
  - Adds force to the particle
- **void ResetForce();**
  - Resets the force
- **void UpdatePosition(float time);**
  - Updates the position of the particle smoothly each frame
- **void UpdateVelocity(float time);**
  - Updates the velocity of the particle
- **void Update(float time);**
  - Main update function used to call UpdatePosition, UpdateVelocity, and ResetForce
- **void CheckLife(float time);**
  - Checks the lifespan of the particle
- **void Destroy();**
  - Toggle particle to destroyed
- **bool IsDestroyed();**
  - Checks if the particle is destroyed

Camera Class:

- The parent camera class for Orthographic and Perspective class

Functions:

- **Camera(float height, float width, Shader* shader)**
  - The constructor of the camera.
  - Stores the height and width of the window
  - Sets the shader the camera will use
- **void Update(glm::vec3 pos, glm::vec3 center);**
  - Updates the camera position
  - Updets the projection matrix
  - Updates the view matrix
- **glm::vec3 getCameraPos();**
  - Returns the camera position
- **glm::mat4 getViewMatrix();**
  - Returns the camera's view matrix
- **glm::mat4 getProjectionMatrix();**
  - Returns the camera's projection matrix
- **void setViewMatrix(glm::vec3 viewFront);**
  - Sets the camera's view matrix
- **void setCameraUse(bool use);**
  - Sets the camera use condition to true or false
  - Used when switching between the two cameras (ortho and perspective)
- **bool getCameraUse();**
  - Returns the camera's use condition
- **void setCameraPos(glm::vec3 cameraPos);**
  - Sets the camera position
- **virtual void setProjectionMatrix() = 0;**
  - A pure virtual function that will set the projection matrix of the camera.

PerspectiveCamera Class;

- Its the perspective camera

Functions:

- **PerspectiveCamera(float height, float width, Shader* shader);**
    - The constructor of the perspective camera.
    - Stores the height and width of the window
    - Sets the shader the camera will use

- **void setProjectionMatrix();**
    - Sets the projection matrix of the perspective camera

Orthographic Camera Class;

- Its the orthographic camera

Functions:

- **Orthographic(float height, float width, Shader* shader);**
    - The constructor of the orthographic camera.
    - Stores the height and width of the window
    - Sets the shader the camera will use

- **void setProjectionMatrix();**
    - Sets the projection matrix of the orthograhpic camera

Shader Class:

- The shader class handles the making of the shader that will be used.
- Also has help functions for glUniforms

Functions:

- **Shader(const char* vertexPath, const char* fragmentPath);**
    - Handles the generation of shader
- **void use()**
    - Activates the shader

HELPER GLUNIFORM FUNCTIONS:

- **void setBool(const std::string& name, bool value) const**
- **void setInt(const std::string& name, int value) const**
- **void setFloat(const std::string& name, float value) const**
- **void setGLfloat(const std::string& name, float value, GLfloat* a) const**
- **void setMat4(const std::string& name, float value, glm::mat4 matrix) const;**

void checkCompileErrors(unsigned int shader, std::string type);

- utility function for checking shader compilation/linking errors

RenderParticle Class;

- Holds both particle and object

Functions:

- **RenderParticle(P6::P6Particle* Particle, Object* object);**
  - The constructor of the renderparticle class
  - Stores the particle and the object
- **void Draw();**
  - If the particle is not destroyed, sets the position of the object equal to the position of the particle
  - Calls the draw function of the object
- **P6::P6Particle* getParticle();**
  - Returns the particle;

ForceGenerator Class

- Parent class for external force generators (drag, gravity…)

Function:

- **virtual void UpdateForce(P6Particle* p, float time)**
  - Adds force to the particle

GravityForceGenerator Class

- The class that generates gravity

Function:

- **GravityForceGenerator(const MyVector Gravity);**
  - The constructor of the class.
  - Sets the gravity
- **void UpdateForce(P6Particle* particle, float time);**
  - Applies gravity force to the particle

DragForceGenerator Class

- The class that generates drag

Function:

- **DragForceGenerator();**
  - Default constructor
- **DragForceGenerator(float _k1, float _k2);**
  - Constructor of the class
  - Sets the first and second coefficients of friction
- **void UpdateForce(P6Particle* particle, float time);**
  - Applies drag to the particle

ForceRegistry Class

- The class that stores particles along with force generators

Functions:

- **void Add(P6Particle* particle, ForceGenerator* generator);**
  - Adds the particle and force generator to the registry list
- **void Remove(P6Particle* particle, ForceGenerator* generator);**
  - Removes the particle and force generator on the registry list
- **void Clear();**
  - Clears the registry list
- **void UpdateForces(float time);**
  - Updates the force applied on the particle

PhysicsWorld Class

- Holds the particles in a particle list
- Does the updates for all particles each second

Functions:

- **void AddParticle(P6Particle* toAdd);**
  - Adds the particle to the particle list and registry list along with gravity
- **void Update(float time);**
  - Calls the update of each particle in the particle list
- **void UpdateParticleList();**
  - Removes particle in the list if the particle is destroyed

MyVector Class

- A helper class of operators
- Acts as the vector3

Functions:

- **MyVector();**
  - The default constructor of the class
- **MyVector(const float _x,**
  const float _y,
  const float _z);
  - The constructor of the class that stores 3 numbers: x, y z

HELPER FUNCTIONS:

- **MyVector operator + (const MyVector v);**
- **MyVector operator - (const MyVector v);**
- **MyVector operator * (const float v);**
- **void operator += (const MyVector v);**
- **void operator -= (const MyVector v);**
- **void operator *= (const float v);**
- **float magnitude();**

- **MyVector direction();**
- **MyVector componentProduct(const glm::vec3 v);**
- **float scalarProduct(const glm::vec3 v);**
- **MyVector vectorProduct(const glm::vec3 v);**