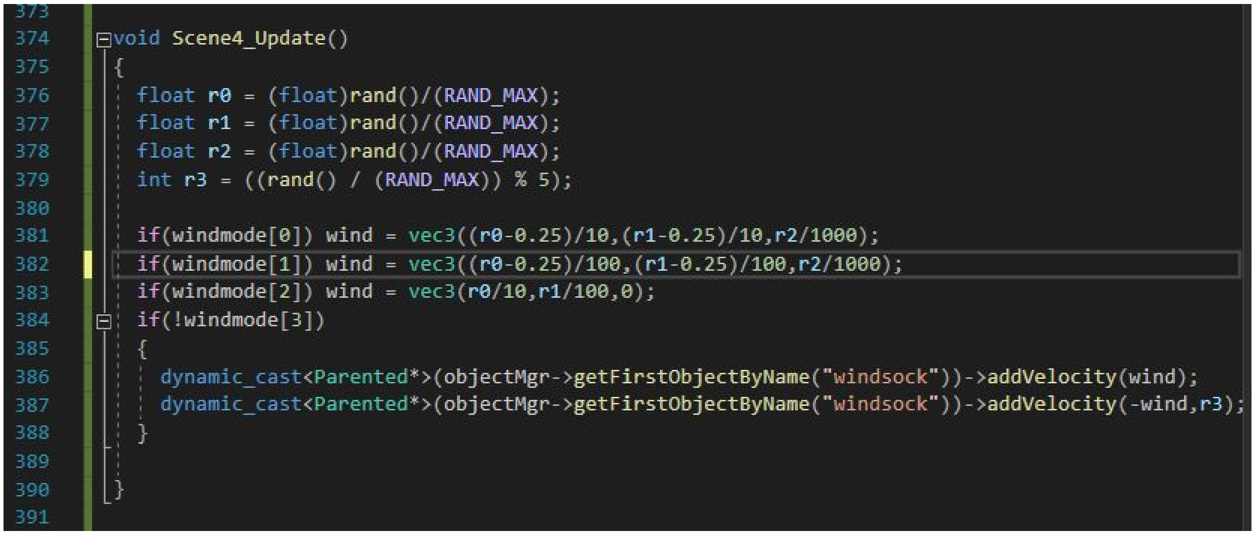
Cody Morgan
CS460
Project 4
2 Dec 2020

Video: https://www.youtube.com/watch?v=WFtNs1mP5Ck

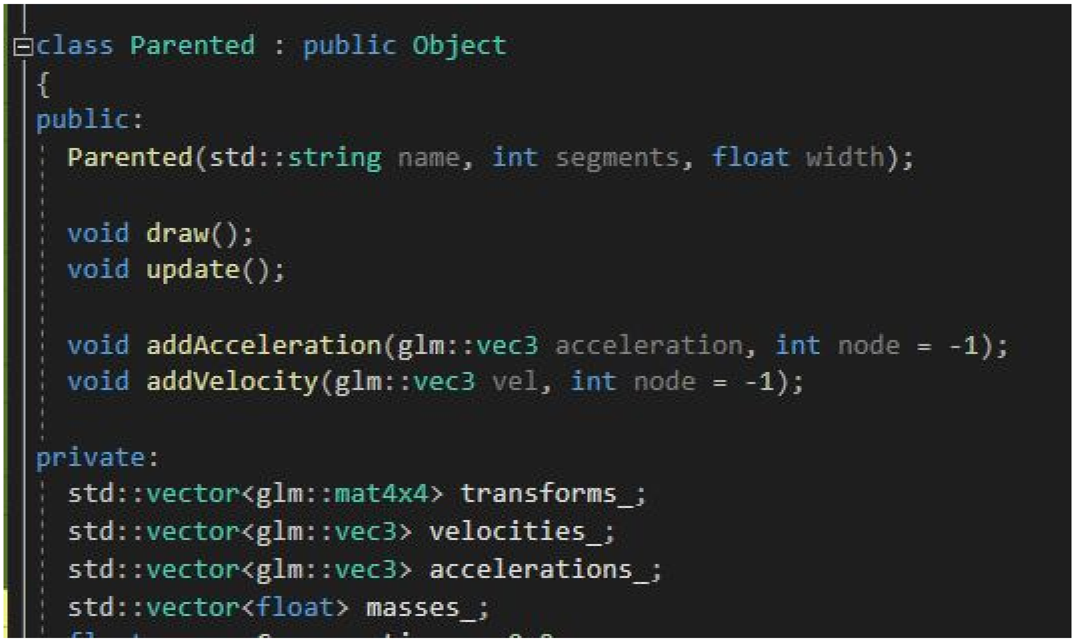
I chose the linked chain simulation for this project. I used Runge-Kutta-4 to simulate a “windsock” blowing in the wind. There is a limited UI that allows for 4 different wind types: hurricane, turbulent, low wind, and no wind.



```
373
374  void Scene4_Update()
375  {
376    float r0 = (float)rand()/(RAND_MAX);
377    float r1 = (float)rand()/(RAND_MAX);
378    float r2 = (float)rand()/(RAND_MAX);
379    int r3 = ((rand() / (RAND_MAX)) % 5);
380
381    if(windmode[0]) wind = vec3((r0-0.25)/10,(r1-0.25)/10,r2/1000);
382    if(windmode[1]) wind = vec3((r0-0.25)/100,(r1-0.25)/100,r2/1000);
383    if(windmode[2]) wind = vec3(r0/10,r1/100,0);
384    if(!windmode[3])
385    {
386      dynamic_cast<Parented*>(objectMgr->getFirstObjectByName("windsock"))->addVelocity(wind);
387      dynamic_cast<Parented*>(objectMgr->getFirstObjectByName("windsock"))->addVelocity(-wind,r3);
388    }
389
390  }
391
```

(Main.cpp line 381)

This linked chain, has the ability to have forces added from the outside (in the case of wind or other forces) as well as acceleration (in the case of gravity or other constant acceleration)

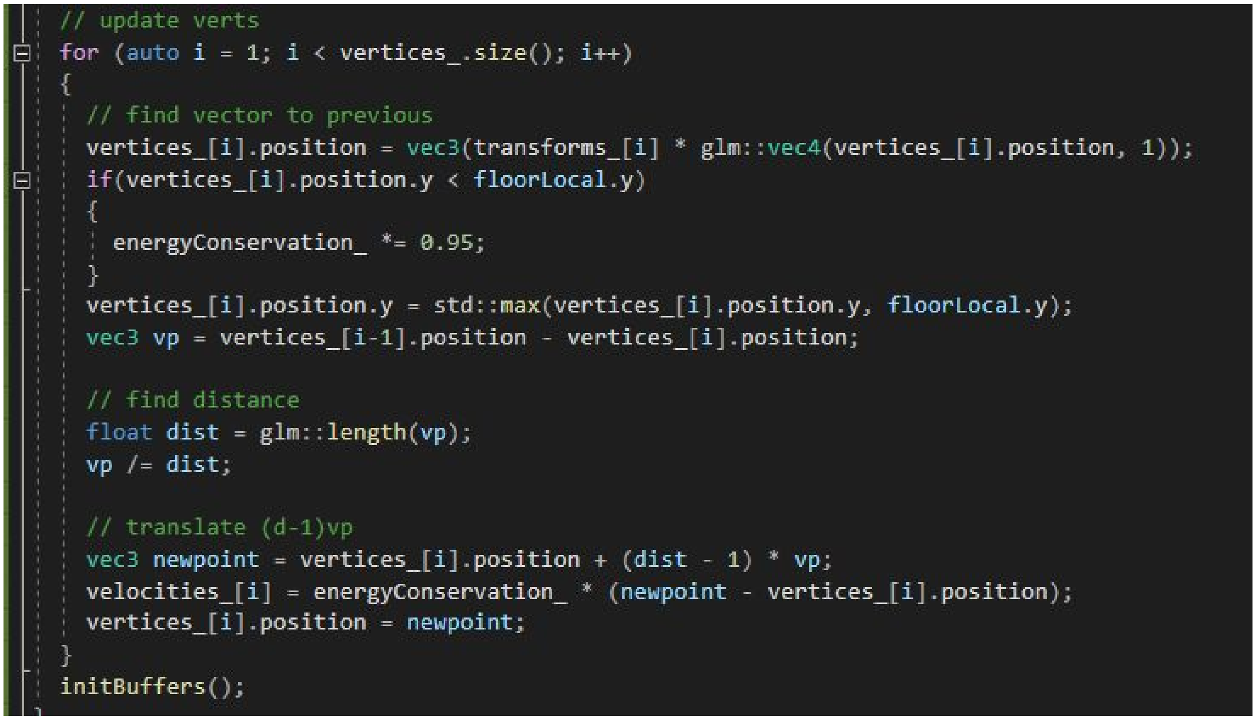


```
class Parented : public Object
{
public:
  Parented(std::string name, int segments, float width);

  void draw();
  void update();

  void addAcceleration(glm::vec3 acceleration, int node = -1);
  void addVelocity(glm::vec3 vel, int node = -1);

private:
  std::vector<glm::mat4x4> transforms_;
  std::vector<glm::vec3> velocities_;
  std::vector<glm::vec3> accelerations_;
  std::vector<float> masses_;
```

(ParentedObject.h)

Cody Morgan
CS460
Project 4
2 Dec 2020

The links in the chain are parented using a similar technique we used for the bones in previous projects.



```
// update verts
for (auto i = 1; i < vertices_.size(); i++)
{
  // find vector to previous
  vertices_[i].position = vec3(transforms_[i] * glm::vec4(vertices_[i].position, 1));
  if(vertices_[i].position.y < floorLocal.y)
  {
    energyConservation_ *= 0.95;
  }
  vertices_[i].position.y = std::max(vertices_[i].position.y, floorLocal.y);
  vec3 vp = vertices_[i-1].position - vertices_[i].position;

  // find distance
  float dist = glm::length(vp);
  vp /= dist;

  // translate (d-1)vp
  vec3 newpoint = vertices_[i].position + (dist - 1) * vp;
  velocities_[i] = energyConservation_ * (newpoint - vertices_[i].position);
  vertices_[i].position = newpoint;
}
initBuffers();
```

(ParentedObject.cpp line 63)