# Graphics 1, Final

By David Zuniga Navarro

## How to build:

1. Project is made and compiled in VS2019.
2. Open the solution file, and make sure configuration is set to x64, Debug/Release.
3. Release is recommended for a better experience.

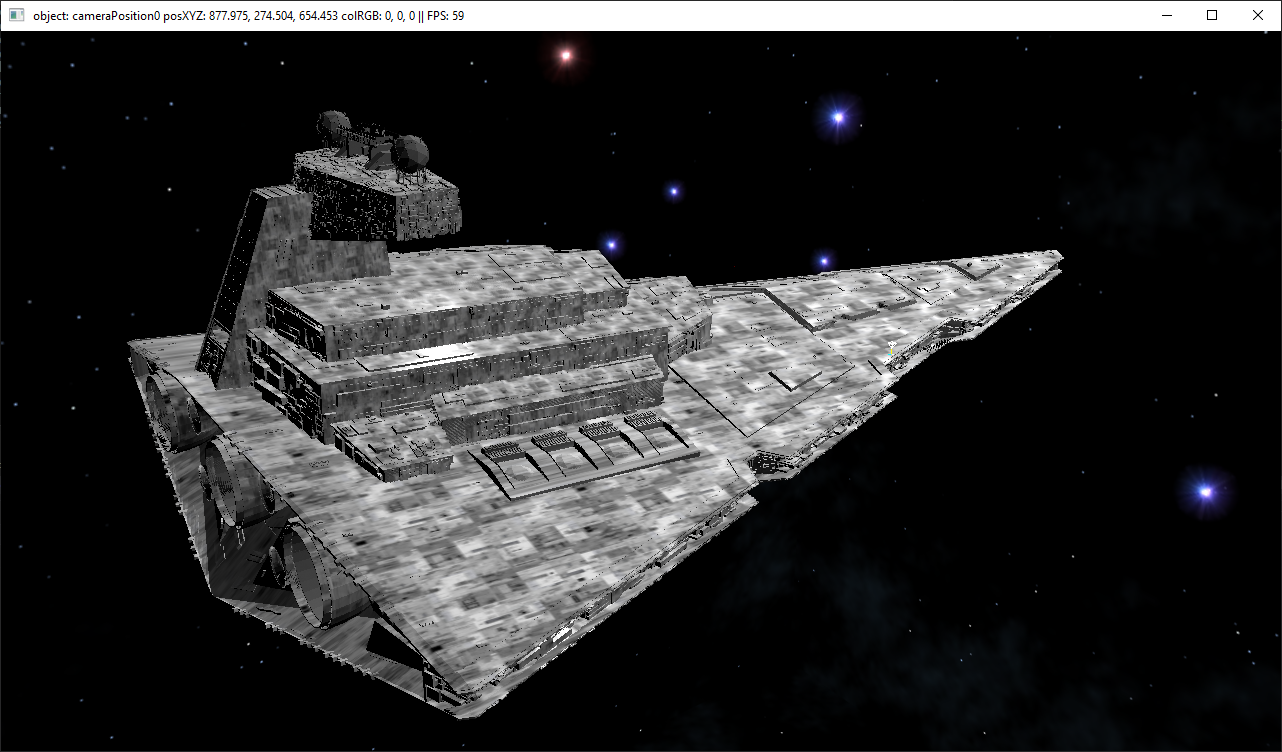
## How to use:

* WASD - Move the camera in X and Z axis
* QE - Move the camera in the Y axis
* Mouse - While left clicking moves the camera
* 2 - Attack Run.
* F2 - Follow or stop following the ship manually
* 9 & 0 - Decrease shield generators health the “hacky” way

## Answers:

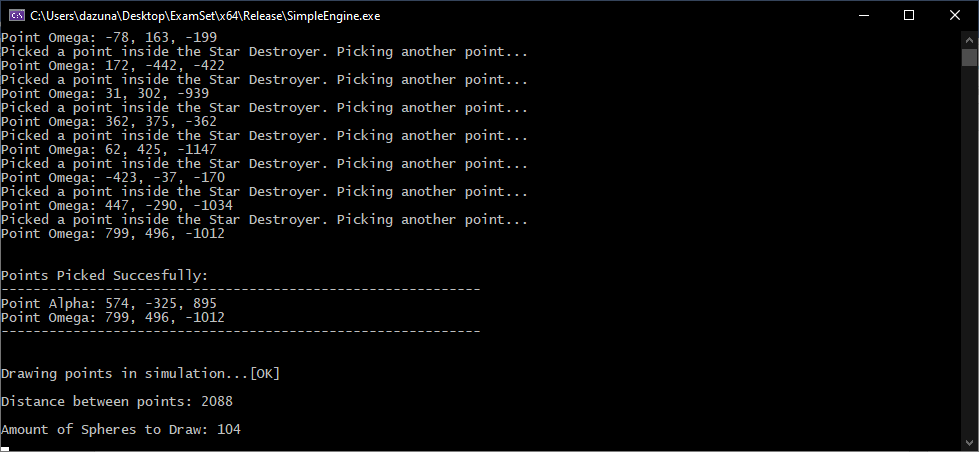
Most of the code can be found in the PhysicsAABBStuff.cpp file

### Question1



### Question2





### Question3

void shootBullet(cGameObject\* xwing)

{

cGameObject\* bullet = ::g\_map\_GameObjects["bullet"];

bullet->positionXYZ = xwing->positionXYZ;

bullet->setAT(glm::normalize(xwing->velocity));

bullet->velocity = xwing->velocity;

//bullet->tag = "lifetime";

//bullet->lifetime = 100.0f;

}

void makeBulletHit(cGameObject\* bullet)

{

duplicateSphere(bullet->positionXYZ, "red", 7.0f, 0.5f, 10000.0f);

bullet->velocity = glm::vec3(0);

bullet->positionXYZ = glm::vec3(10000, 0, 0);

}

Video attached “question3”

### Question4

void shootBullet(cGameObject\* xwing)

{

cGameObject\* bullet = ::g\_map\_GameObjects["bullet"];

bullet->positionXYZ = xwing->positionXYZ;

bullet->setAT(glm::normalize(xwing->velocity));

bullet->velocity = xwing->velocity;

//bullet->tag = "lifetime";

//bullet->lifetime = 100.0f;

}

void makeBulletHit(cGameObject\* bullet)

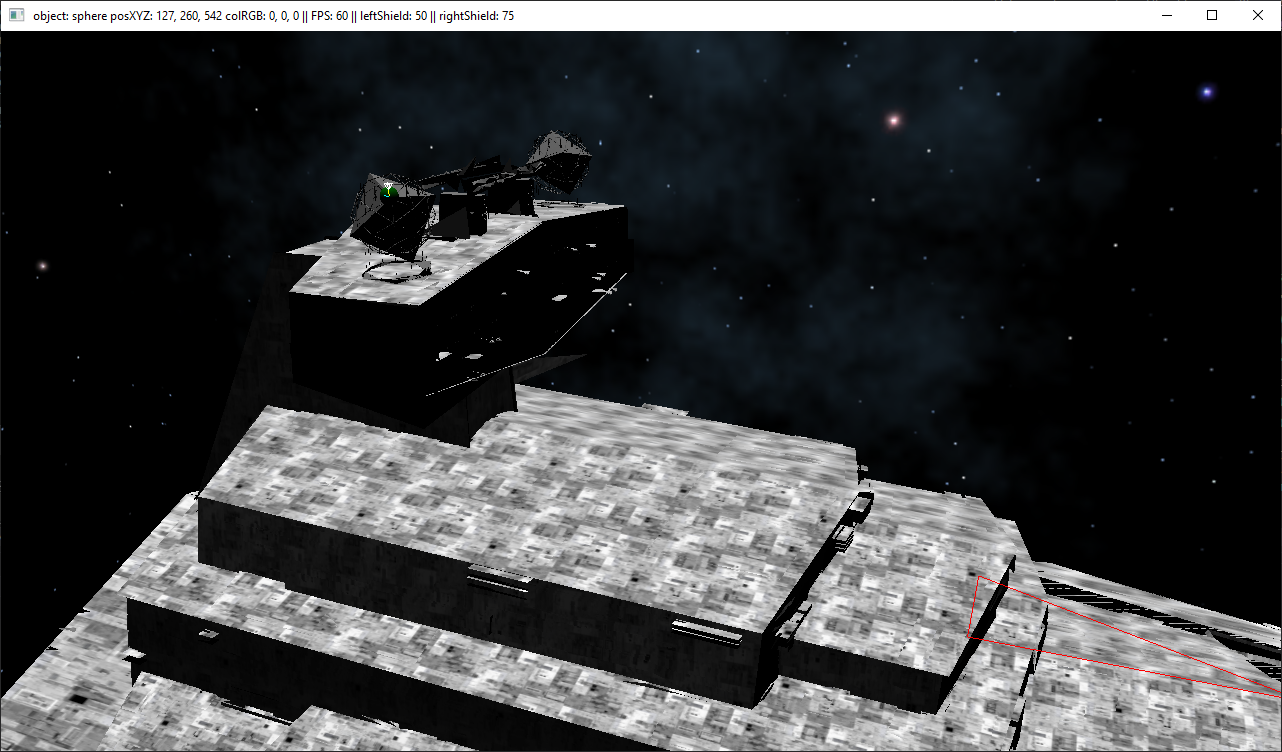
{

duplicateSphere(bullet->positionXYZ, "red", 7.0f, 0.5f, 10000.0f);

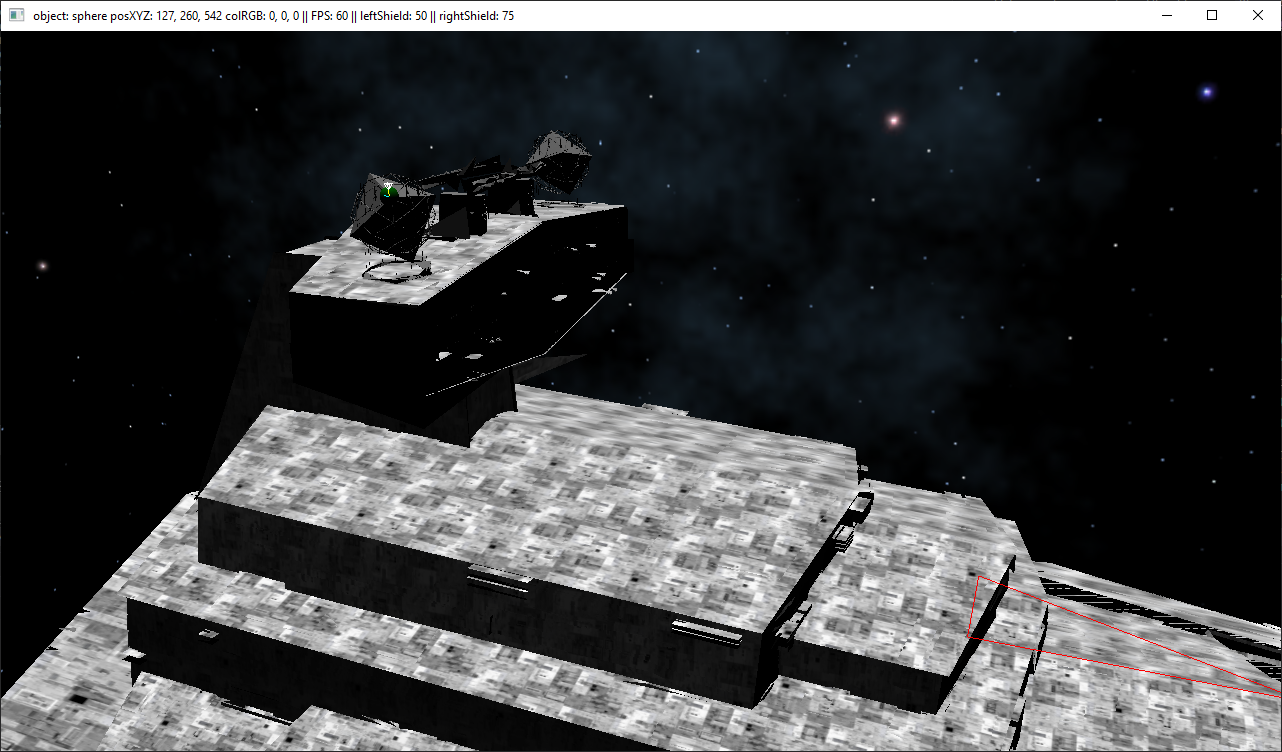
bullet->velocity = glm::vec3(0);

bullet->positionXYZ = glm::vec3(10000, 0, 0);

}



### Question4



void makeBulletHit(cGameObject\* bullet)

{

duplicateSphere(bullet->positionXYZ, "red", 7.0f, 0.5f, 10000.0f);

// validate the hit was to the Shield Generator

glm::vec3 leftGenerator = glm::vec3(101, 245, 556);

glm::vec3 rightGenerator = glm::vec3(-101, 245, 556);

float distLeft = glm::distance(leftGenerator, bullet->positionXYZ);

float distRight = glm::distance(rightGenerator, bullet->positionXYZ);

if (distLeft <= 200.0f)

{

decreaseLife("left");

}

if (distRight <= 200.0f)

{

decreaseLife("right");

}

bullet->velocity = glm::vec3(0);

bullet->positionXYZ = glm::vec3(10000, 0, 0);

}

This snippet validates that the bullet hit the shield generators given that at this point in the code, collision has been detected by the aabb physics. We just check if this collision was with the spheres that are the shield generators.

And boom goes the dynamite.

### Question5

Attached video: “question5”

VERTEX SHADER:

uniform bool itsDeadJim;

uniform float offset;

void main()

{

vec4 vertPosition = vPosition;

mat4 matMVP = matProj \* matView \* matModel;

if(itsDeadJim)

{

vertPosition.xyz \*= (vNormal.xyz \* offset);

}

gl\_Position = matMVP \* vec4(vertPosition.xyz, 1.0);

Validation In code:

void shouldStarDestroyerExplode(float averageDeltaTime)

{

float totalHealth = leftShieldHealth + rightShieldHealth;

if (totalHealth <= 0.0f)

{

itsDeadJim = true;

offset += averageDeltaTime;

}

}

NOTE:

Making the star destroyer explode can be done by: pressing [2] until both Generators are hit 4 times each, which takes WAY too long.

OR

Keys [9] and [0] decrease the life of the left and right generators which lets us get to the big boom faster.