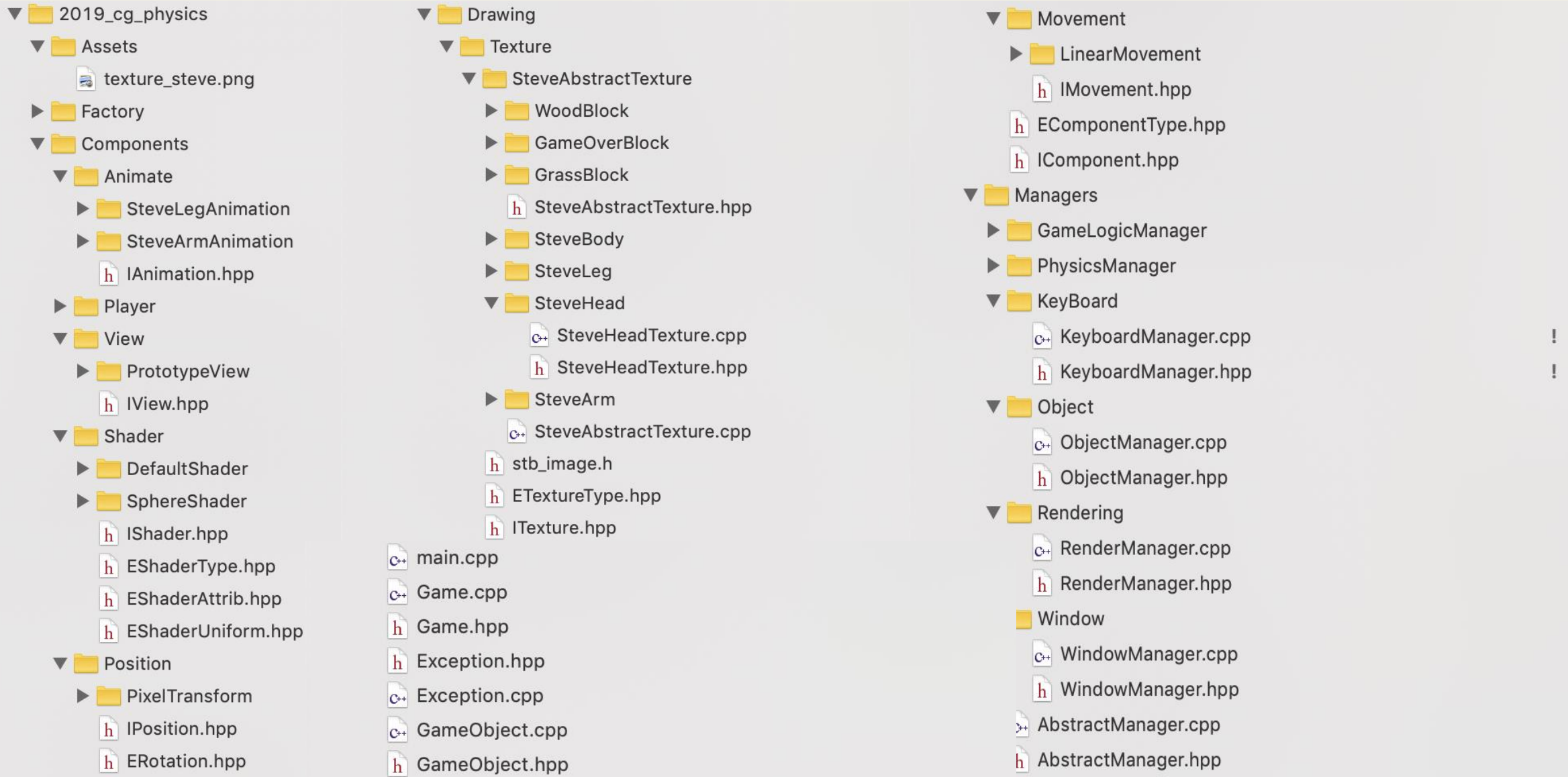
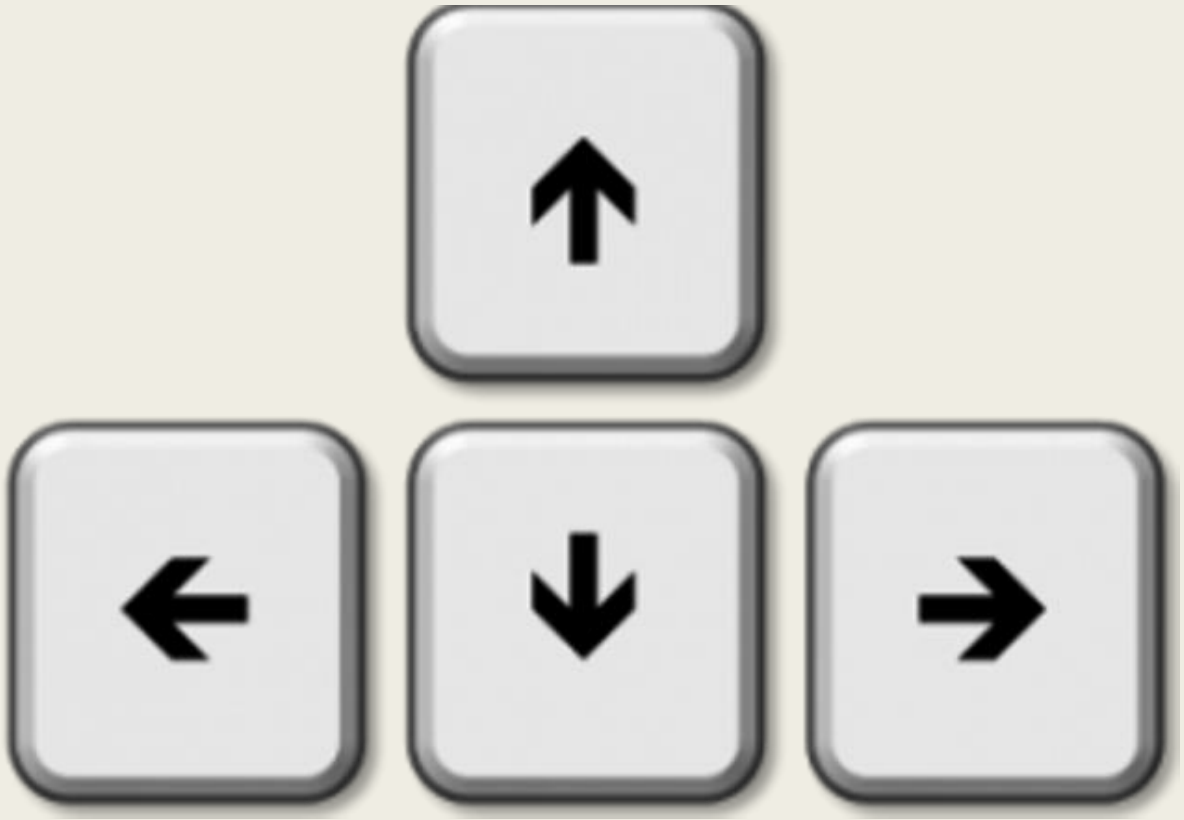
MINECRAFT BALLS

Spiel Einführung.

Framework


2019_cg_physics
Assets
texture_steve.png
Factory
Components
Animate
SteveLegAnimation
SteveArmAnimation
IAnimation.hpp
Player
View
PrototypeView
IView.hpp
Shader
DefaultShader
SphereShader
IShader.hpp
EShaderType.hpp
EShaderAttrib.hpp
EShaderUniform.hpp
Position
PixelTransform
IPosition.hpp
ERotation.hpp
Drawing
Texture
SteveAbstractTexture
WoodBlock
GameOverBlock
GrassBlock
SteveAbstractTexture.hpp
SteveBody
SteveLeg
SteveHead
SteveHeadTexture.cpp
SteveHeadTexture.hpp
SteveArm
SteveAbstractTexture.cpp
stb_image.h
ETextureType.hpp
ITexture.hpp
main.cpp
Game.cpp
Game.hpp
Exception.hpp
Exception.cpp
GameObject.cpp
GameObject.hpp
Movement
LinearMovement
IMovement.hpp
EComponentType.hpp
IComponent.hpp
Managers
GameLogicManager
PhysicsManager
KeyBoard
KeyboardManager.cpp
KeyboardManager.hpp
Object
ObjectManager.cpp
ObjectManager.hpp
Rendering
RenderManager.cpp
RenderManager.hpp
Window
WindowManager.cpp
WindowManager.hpp
AbstractManager.cpp
AbstractManager.hpp



STEUERUNG

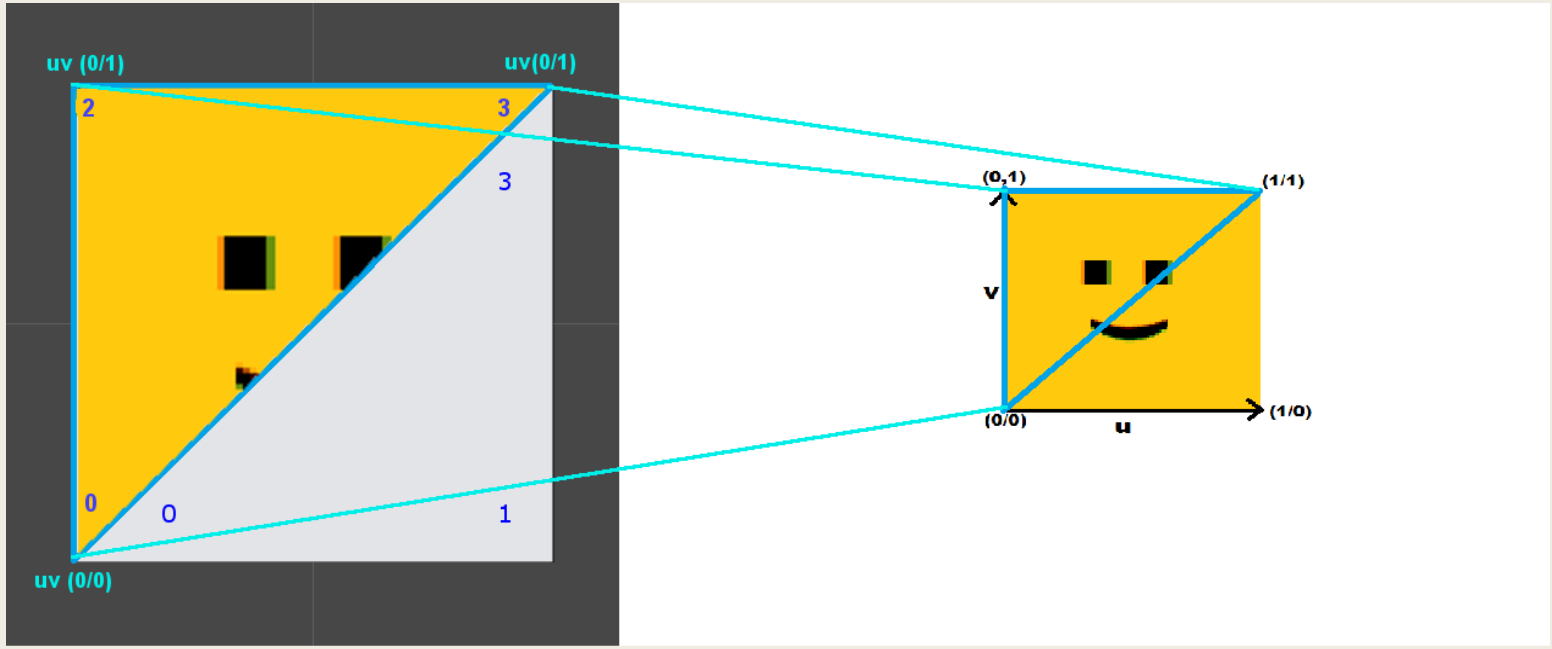
```
void KeyboardManager:: keyCallbackM(GLFWwindow* myWindow, int key, int scanCode, int action, int mod)
{
    if (((key == GLFW_KEY_ESCAPE) || (key == GLFW_KEY_Q))  &&
        (action == GLFW_PRESS))
    /* close window upon hitting the escape key or Q/q */
        glfwSetWindowShouldClose(myWindow, GL_TRUE);
    else if ((key == GLFW_KEY_RIGHT) && (action == GLFW_PRESS || GLFW_REPEAT== action)) {
        _dir = 0;
    } else if ((key == GLFW_KEY_LEFT) && (action == GLFW_PRESS|| GLFW_REPEAT== action)) {
        _dir = 1;
    } else if ((key == GLFW_KEY_DOWN) && (action == GLFW_PRESS || GLFW_REPEAT== action)) {
        _dir = 2;
    } else if ((key == GLFW_KEY_UP) && (action == GLFW_PRESS || GLFW_REPEAT== action)){
        _dir = 3;
    } else {
        _dir = -1;
    }
}
```

```
auto player = dynamic_cast<IPlayer*>((*it)->GetComponent(EComponentType::Player));
auto movement = dynamic_cast<IMovement*>((*it)->GetComponent(EComponentType::Movement));

if(player != nullptr && movement != nullptr) {
    //_dir = GetMoveDirection();
   // std::cout << _dir<< "\n";
    Vector3 velocity = Vector3(0.0, 0.0, 0.0);
    if(_dir == 0) velocity = Vector3(1.0, 0.0, 0.0);
    else if(_dir == 1) velocity = Vector3(-1.0, 0.0, 0.0);
    else if(_dir == 2) velocity = Vector3(0.0, 0.0, 1.0);
    else if(_dir == 3) velocity = Vector3(0.0, 0.0, -1.0);

    movement->SetVelocity(velocity);
}
```

Texture

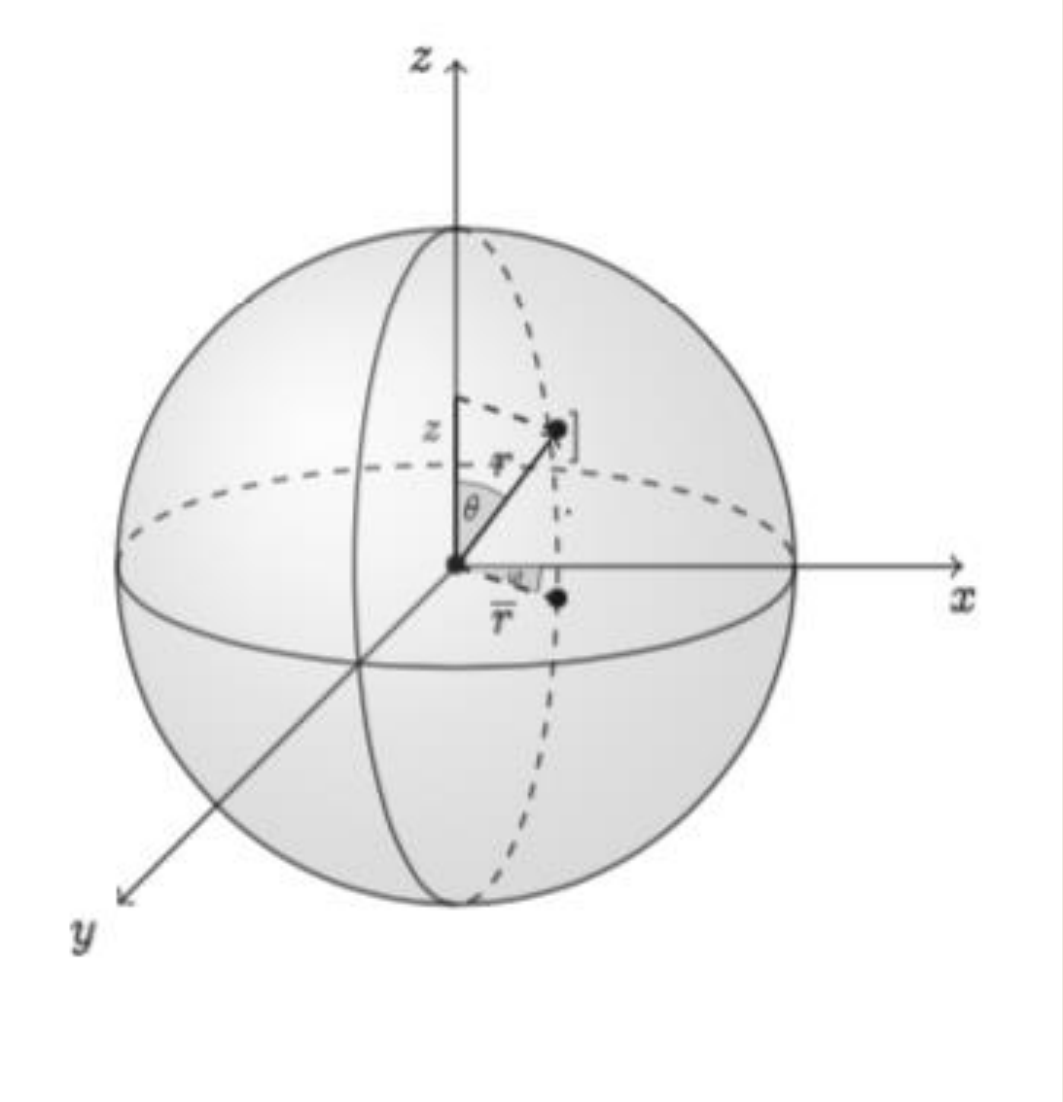


UV COORDINATEN

Sind zwischen 0.0 und 1.0

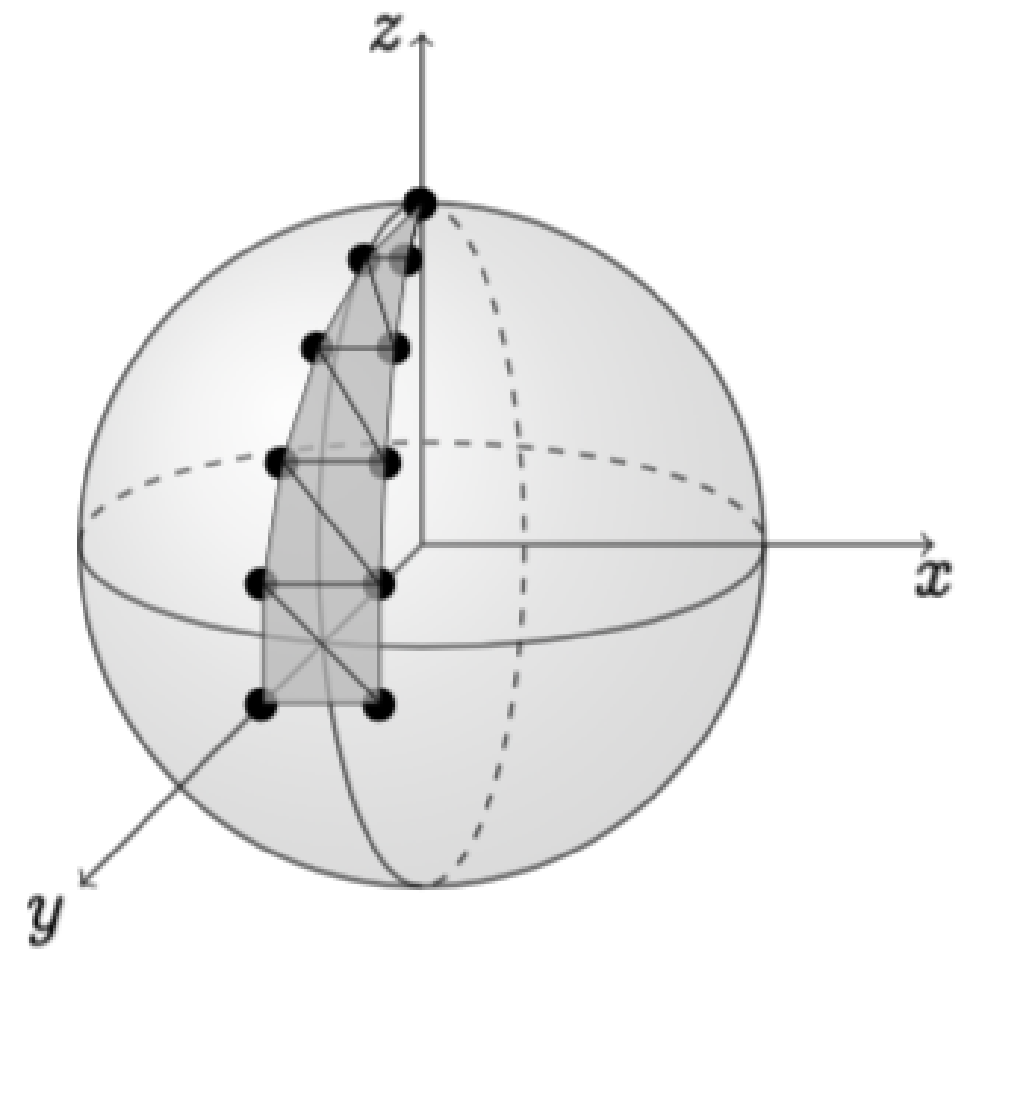

game over
game over

Scenen Graph Fri



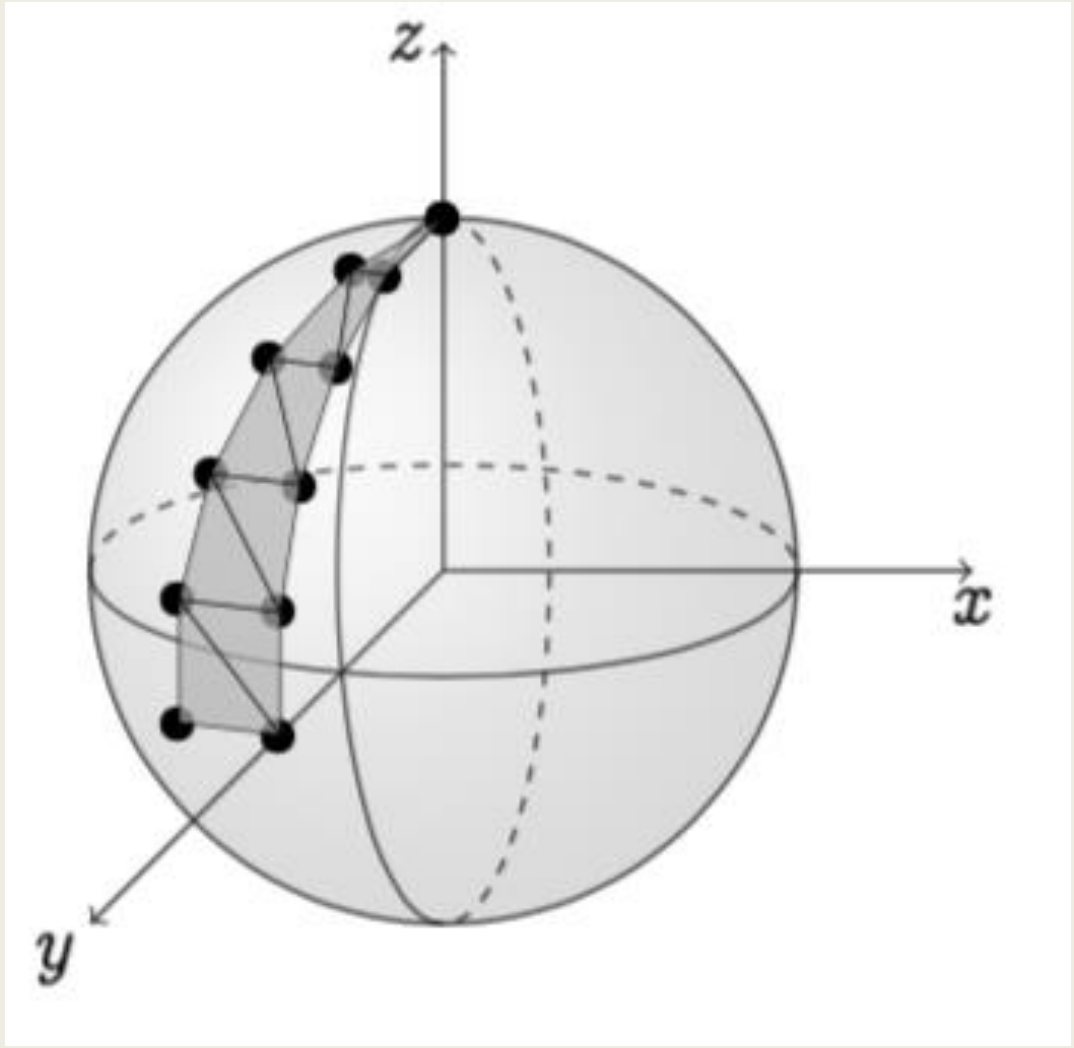
Kartesisches Koordinatensystem

$z =$	$r \cdot \cos\Theta$
$x =$	$r \cdot \sin\Theta \cdot \cos\varphi$
$y =$	$r \cdot \sin\Theta \cdot \sin\varphi$



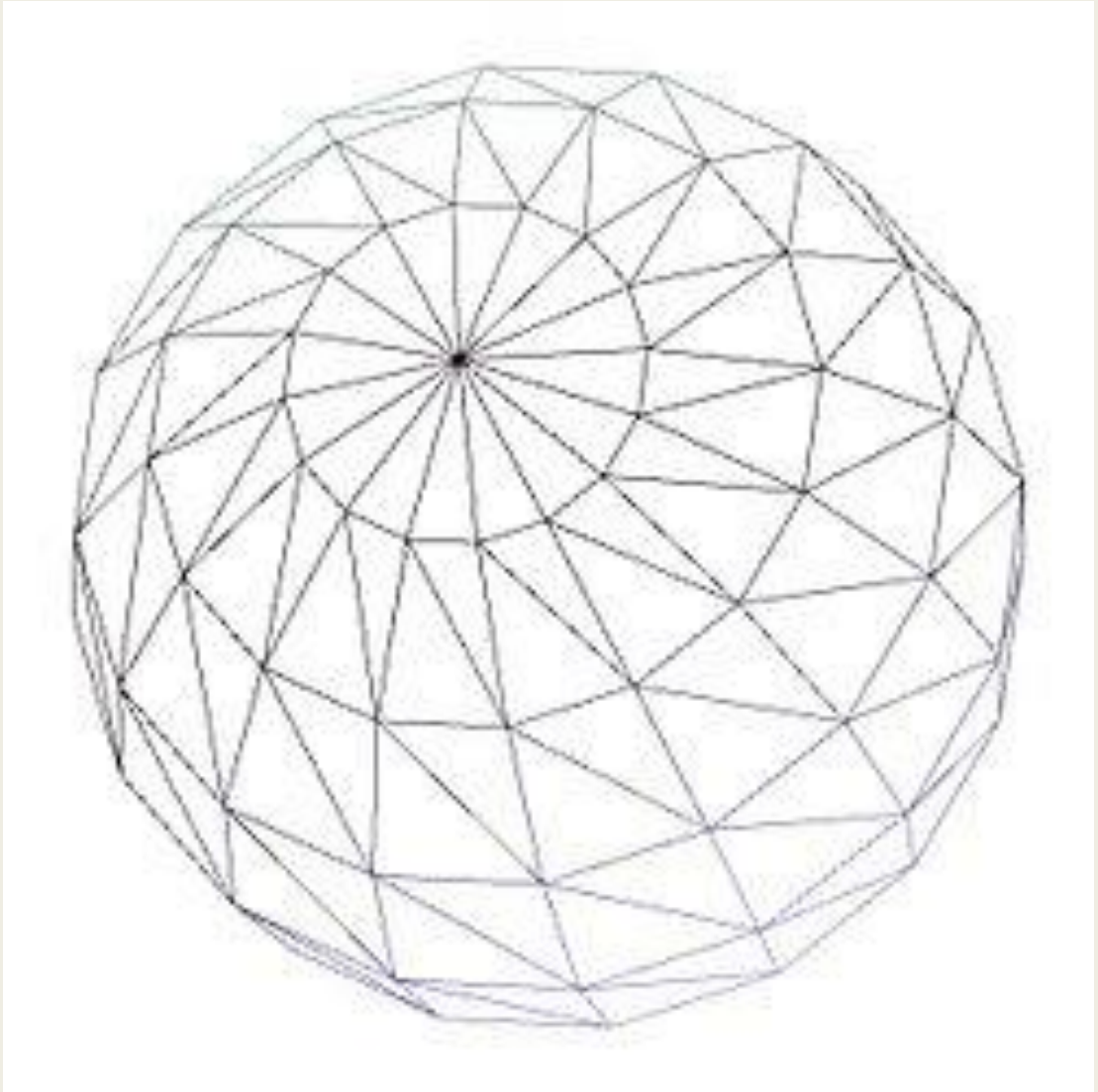
Triangularisierung

- Θ =0,20,40,60,80,100 …360;
- Φ =80,100;

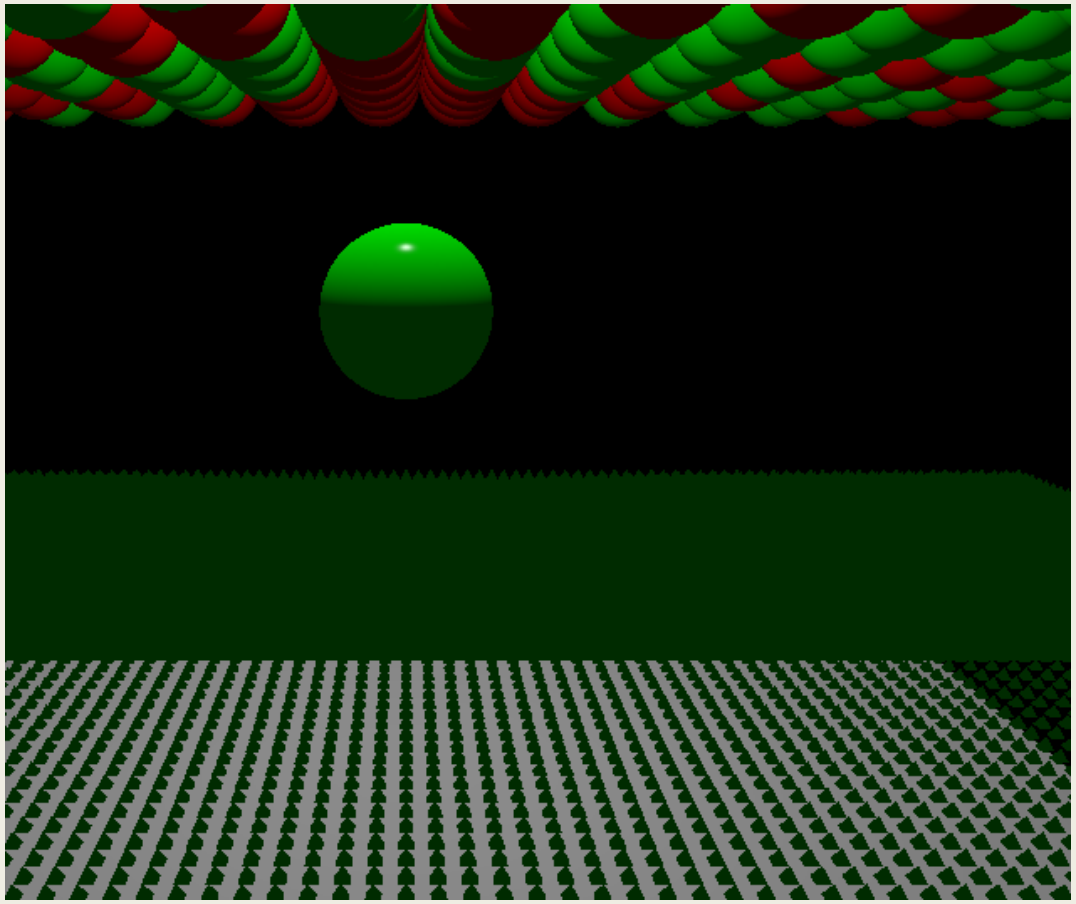


Triangularisierung

- Θ =0,20,40,60,80,100 …360;
- Φ =100,120;



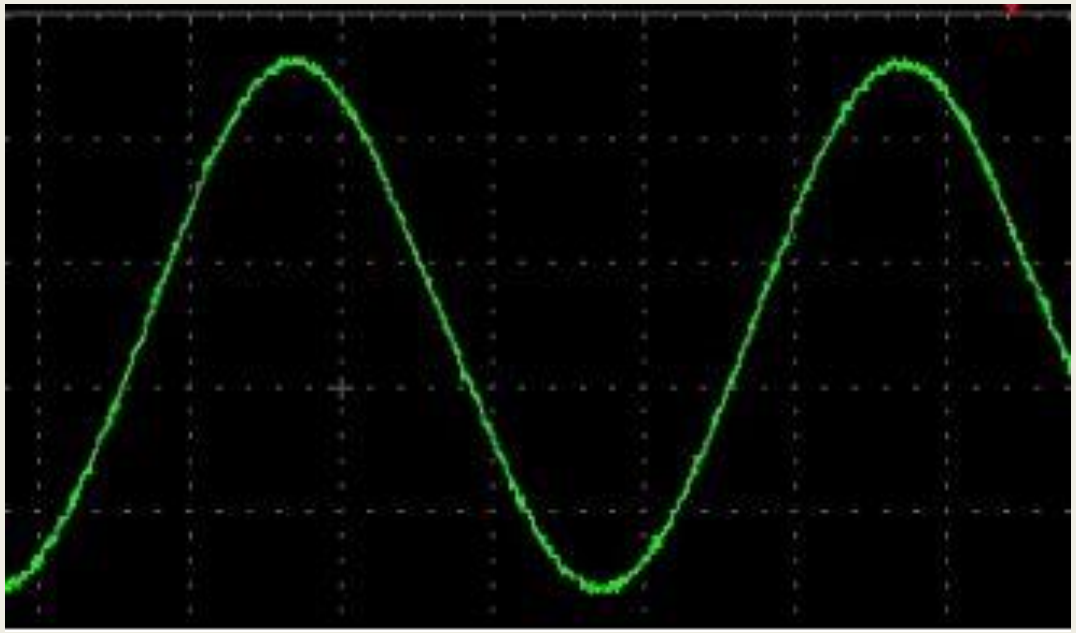
Triangularisierung



Physic

- y = -speed*(timeEnd -timeStart);
- x =currentPos.x;
- z =currentPos.z;

Physic



- x =(+/-)speed*(time)/per;
- y = std::abs(amp * sinf(speed * (time)))
- z =currentPos.z;