

# Features

## Gameplay

---

### 3D Geometry (6 Points)

Non-trivial 3D objects are created using blender and exported as .obj files.

These .obj files are used to create a `MyPhysxDynamicGeometryData` or a `MyPhysxStaticGeometryData` which are structures created in the `MyPhysx` class. These structures contain the VAO which is rendered by the shader program and the PhysX actor object which is used for collision detection.

The VAOs are generated using the `MyVaoBuilder` class. There the .obj file is read and the vertex position, the normals, the indices, the tangent vectors and the UV texture coordinates are saved in a VAO.

In the `MyPhysx` class a PhysX object is created from the vertex positions and the indices using the PhysX cooking library. Some methodes to create objects there are:

```
MyPhysxStaticGeometryData MyPhysx::createStaticCube(...)
MyPhysxStaticGeometryData MyPhysx::createStaticSphere(...)
MyPhysxStaticGeometryData MyPhysx::createStaticConvex(...)
MyPhysxStaticGeometryData MyPhysx::createStaticTriangleMesh (...)
```

```
MyPhysxDynamicGeometryData MyPhysx::createDynamicCube(...)
MyPhysxDynamicGeometryData MyPhysx::createDynamicSphere(...)
MyPhysxDynamicGeometryData MyPhysx::createDynamicConvex(...)
```

### Playable (3 Points)

You control a golf ball and must sink it in the hole of the golf course to win.

### Advanced Gameplay (3 Points)

The game has three levels, and you can choose the level where you want to start. The levels are mini golf courses, where you must sink the ball within 2 minutes. If you need longer, you lose the game, get a game over and the game closes. If you win a level you get to the next on by pressing ENTER, when you won all three levels you won the game.

### Min. 60 FPS and Framerate Independence (3 Points)

The game runs on my PC with 144 FPS in windowed mode and with the 60 FPS set by the framerate value in Fullscreen mode.

Frame independency is achieved with the delta time value. It is used to control the camera speed, the power change and it is used in the PhysX scene simulate method to make the game physics frame independent.

### Win/Lose Condition (3 Points)

You win the game by sinking the golf ball in under 2 minutes in the hole.

You lose the game if you do not sink the ball in 2 minutes.

## Intuitive Controls (2 Points)

Controls for the game:

**1 2 3** – Switch between levels fast (debug use)

**0** – Win current level (debug use)

**UP** – Up in the menu

**DOWN** – Down in the menu

**W** – more Power, Up in the menu

**S** – less Power, Down in the menu

**F** – shoot the golf ball with the set power

**P** – pause the level

**SPACE** – jump with the ball

**Left Mouse Button** – when clicked on the game you can drag the camera around the ball

**Scrolling** – to change the distance between camera and the ball

**F1** – Toggle wireframe representation

**F2** – Toggle face culling

**F11** – Toggle Fullscreen mode

## Intuitive Camera (2 Points)

The game uses an Arc ball camera, modified by mouse input.

The class for the camera is the MyCamera class.

To better see the shooting direction, there is a triangle in front of the golf ball pointing in the direction where the ball will get shot. The triangle length also changes with the shooting power.

## Illumination Model (2 Points)

There is one directional light and multiple point lights in each level.

Material values are set to the shader manually before each draw call when they should get changed. For textures and materials there are separate classes implemented.

Each object has generated or read from the .obj normals stored in the VAO.

## Textures (2 Points)

Every object must have a texture and material to be rendered.

## Moving Objects (2 Points)

The windmill wings are rotating.

## Adjustable Parameters (1 Point)

The following parameters can be adjustable without recompiling the application by changing the settings.ini in the assets folder:

- Screen Resolution (Width/Height)
- Fullscreen-Mode (On/Off)
- Refresh-Rate
- Brightness

---

## Gameplay (optional)

### Collision Detection (Basic Physics) (4 Points)

Collision detection is covered by the PhysX library. The initialization and PhysX methods are implemented in the MyPhysx class.

### Advanced Physics (6 Points)

In the Level 2 of the game there are multiple cube pyramid towers which can be shot down with the golf ball.

### Heads-Up Display (4 Points)

The HUD consists of two elements. One part is the rendered Text in the top left corner. It provides information over the stroke count, the time left for the level and the FPS. The other part of the HUD is the power bar to better visualise the shooting power amount.

For rendering of the text the freetype library is used. The implementation of the text rendering is described in the class Text\_renderer. In the Load method the first 255 ASCII characters are generated and stored for later use. The method RenderText is responsible for the rendering of actual text. It takes the text, the position of the text, the text size and the text colour as input values. Then these values are passed to the associated shader using uniforms.

The power bar is made of two 2d rectangles. The first one has a dynamic size and a dynamic colour, which change dependent on the shooting power change through the keyboard.

In the GameHUD class the loading screen, the win/lose messages and the menu are also implemented.

---

## Gameplay (optional)

### Advanced Modelling:

#### CPU Particle System (8 Points)

Used to generate the particles behind the golf ball when shot.

## Animation

### Hierarchical Animation (4 Points)

The windmill and windmill wings are stored in a struct object created in the Levelmanager class. The wings are a separate mesh which gets rotated. On the rotating wings are four smaller wings which rotate twice as fast in the other direction.

### Vertex Shader Animation (8 Points)

The water has simple waves implemented, the calculations are made in the vertex shader.

## Texturing

### Specular Map (4 Points)

Used from the golf ball.

### Environment Map (8 Points)

Using a skybox cubemap around the golf course to get a background to the game. Also used by some objects like the golf ball, trees or windmills to create reflections.

## Shading

### Simple Normal Mapping (4 Points)

Normal vectors are read from a texture for the golf ball. They are transferred to the tangent space, by using the tangent vectors created by the creation of the VAO of the sphere.

## Post Processing

### Contours via Backfaces (4 Points)

The contours for the golf ball, the rocks and cubes are generated by the slightly larger contour Vao backfaces which are created in the MyPhysx class.

## Additional libraries

opengl32.lib

- glew32s.lib
- glfw3.lib
- ECG\_Library\_Debug.lib

PhysX: (<https://github.com/NVIDIAGameWorks/PhysX>)

- PhysXPvdSDK\_static\_32.lib
- PhysXCommon\_32.lib
- PhysXExtensions\_static\_32.lib
- PhysXFoundation\_32.lib
- PhysX\_32.lib
- PhysXCooking\_32.lib

FreeType: ()

- freetype.lib