# **EZG-Project**

# Controls

|  |  |
| --- | --- |
| **Free camera control** | |
| W | Move forward |
| A | Move left |
| S | Move backward |
| D | Move right |
| Mouse X | Control camera yaw |
| Mouse Y | Control camera pitch |
| Q | Rotate camera roll anticlockwise |
| E | Rotate camera roll clockwise |
| F1 | Reset camera roll |
| **Spline control** | |
| F2 | Select previous way point |
| F3 | Select next way point |
| F4 | Toggle free-cam/spline-mode |
| space | Add new way point |
| delete | Delete selected way point |
| enter | Print way points |
| **other** | |
| Comma | Reduce speed |
| Period | Enhance speed |
| Shift comma | Reduce bumpiness |
| Shift period | Enhance bumpiness |
| Strg comma | Reduce sample count (antialiasing) |
| Strg period | Enhance sample count (antialiasing) |
| Esc | Quit game |

# Important program segments

|  |  |  |
| --- | --- | --- |
| main() | Program initialisation, creation of objects, shaders, framebuffers…. , contains main-loop | main.cpp – line 92 |
| main() | Camera position in spline mode is set correctly (even speed) | main.cpp – line 410 (main-loop) |
| Spline::move() | Shifts the pivot along the spline | Spline.cpp – line 128 |
| Spline::GetSplinePointLocation() | Returns pivot location | Spline.cpp – line 166 |
| Spline::GetSplinePointRotation() | Returns pivot rotation | Spline.cpp – line 201 |
| processInput() | controls | main.cpp – line 573 |
| main() | Set up matrices for light + depth buffer, render to depth buffer | main.cpp – line 441 (main-loop) |
| main() | Set up camera matrices and Co, render scene | main.cpp – line 465 (main-loop) |
| main() | Render kdTree | main.cpp – line 484 (main-loop) |

Further important stuff:

Spline.cpp contains all the code relevant to the spline, minus some small segment at the beginning of the main loop.

Depth\_VS.glsl and Empty\_FS.glsl contain the shaders to render the shadow map.

ShadowMap\_VS.glsl and ShadowMap\_FS.glsl contain the shaders to render the scene with shadows as well as lightning and normal map handling.

The WorldObject class represents an object visible in the world. It has a transform (inherited) and contains a model which in turn stores vertex data as well as textures. When rendering the transform of the WorldObject is used to create a model matrix which is passed to the shader for rendering the model

The main.cpp contains a method called setSamples(). This method is used to recreate a render buffer for antialiasing