

# Whish in the Wind Documentation

## Controls:

- **W:** Move forward
- **A:** Strafe left
- **S:** Strafe right
- **D:** Move backwards

(W,A,S,D movement only accounts for the horizontal (Y axis) camera rotation and thus this keys only affect movement in the x, z plane)

- **Shift:** Move faster
- **Left Ctrl:** Move down
- **Space:** Move up
- **R:** reset game
- **H:** toggle HUD

All movement keys can be combined to create combined movements for example Shift + W + A + Ctrl To move fast diagonally forward, left, down. Opposite movement keys (W and S, A and D, Ctrl and Space) cancel each other out. The movement speed isn't faster by pressing multiple direction keys at once.

- **Esc:** Quit game
- **+:** Increase Illumination multiplier
- **-:** Decrease Illumination multiplier
- **F1:** Toggle wire-frame mode
- **F2:** Toggle back-face culling
- **F3:** Toggle normal mapping
- **F8:** Toggle fullscreen

- **Mouse:** Move the mouse to rotate the camera around the players current position to change the viewdirection (first person camera). The vertical rotation is limited (-89 to 89 Degrees) to prevent the player from doing confusing summersaults or hanging upsidedown.
- **Mouse left:** Create global wind in view direction – all active flying lanterns on the map are accelerated in the viewdirection until the max speed is reached. The longer the mousebutton is hold the longer the acceleration vector becomes until a certain max length is reached.
- **Mouse right:** Create local wind around the player – all active nearby flying lanterns are accelerated away from the player. The acceleration vector is calculated by subtracting the players position from the lanterns position and correcting the length of the vector.

## Implementation

The game is build for 64 Bit machines and uses OpenGL 4.5.

## Features:

## Gameplay:

### Compulsory

**3D Geometry (6 Points):** lanterns and objects are loaded using assimp, terrain is generated with a heightmap and tessellation

**Playable (3 Points):** basic functionality is implemented

**Advanced Gameplay (3 Points):** Lanterns always return to their starting height; lanterns are floaty and do not stop immediately after being hit by wind. If Lantern is destroyed by collision it falls to

the ground where it stops. Local wind only affects Lanterns in a cylinder in front of player. Global wind affects all Lanterns.

**Min 60 FPS and Framerate Independence (3 Points):** Runs at +120 FPS on PC at home (Specs in readme), hard drops if wireframe is enabled and the player looks at the corner in direction (1, -1, 1) in combination with a big terrain., but is fine if player looks at any other direction. Cannot find the reason for this.

**Win/Loose Condition (3 Points):** If all lanterns get destroyed the player loses and a very dramatic ending appears (Video Texture). If at least one reaches the goal (round platform) the player wins, and the amount of saved lantern is shown

**Intuitive Controls (2 Points):** (Almost) Implemented (Most/All Controls are hooked up)

Update: simple wind force is implemented in compute shader

**Intuitive Camera (2 Points):** Implemented. As a bonus we added Gamepad support – trigger control height (at the moment since we might want to use them for finer wind control) and left analogstick movement and right analogstick the camera direction, A to move faster. You can even combine gamepad and mouse and keyboard input. Listening for gamepad connection call-backs makes sure to process such input only when one is connected. Performance is also slightly increased by checking first if opposite keys are in the same state and if not compare only one of the keys to decide. A direction vector is created which together with the players position is used to create a lookat matrix. Gamepad support got neglected as development went on, but would be easy to re-implement

**Illumination Model (2 Points):** Point light got added to every lantern. There is also static light from the lightmaps.

**Textures (2 Points):** Implemented for testing including mirrored tiling and trilinear filtering with mipmaps. We use a sampler object which we can bind to texture units to apply the settings easily to multiple texture units.

**Moving Objects (2 Points):** lanterns move in the wind.

**Documentation (1 Point):** In the works (this one)

**Adjustable Parameters (1 Point):** Implemented (hopefully the way it was intended – waiting for a tuwel Forum answer) Fullscreen uses the resolution and windowed mode the window size.

Brightness just multiplies the resulting color for output with the current `_illuminationMultiplier`. Brightness and Fullscreen can be manipulated in real time with keypresses.

### **Optional:**

**Collision Detection (4 Points):** collision between lanterns and ground implemented, player has no collision because of gameplay reasons (to get a lantern close to a wall unstuck)

**Advanced Physics (6 Points):** Lanterns have collision with other lanterns. Invisible Barrier on top that prevents players from just flying over the map and avoiding all obstacles.

**Heads-Up Display (4 Points):** lantern counter (top right) and wind bar (bottom) that keeps track of how much wind the player can use is implemented as well as a record of how many lanterns were saved (top left). The number textures are not ideal and do not scale well with screen size, if time is left change them to prettier ones.

### **Effects:**

**Lightning: Lightmap using separate textures (8 Points):** Lanterns glow using a lightmap (besides point lights). The flickering of the light is intended to simulate a burning flame inside the lantern. (The effect can be seen best if a single lantern gets isolated from the flock)

**Advanced modelling: GPU Particle System using Transform Feedback (12 Points):** changed after feedback talk one to particle system using computeShader. Is used to calculate lantern positions and physics as well as keeping track of how many are alive.

**Shading: Simple normal mapping (4 Points):** not implemented yet since there are also no objects in the game suitable for normal mapping.

**Post Processing: Bloom/Glow (8 Points):** bright lights aka lanterns glow

### **Optional Effects:**

**Texturing: Video Texture (8 Points):** Can be seen when all lanterns get destroyed. Very poetic and romantic clip. Caped to 24 FPS and a bit over 3 seconds long.

**Terrain: Tessellation from heightmap (12 Points):** fully implemented, tessellation level gets decided on distance of player to two control points, rgb is used for color, alpha channel is used to calculate displacement.

### **Used external Librarys:**

- **glew (version 2.1.0, <http://glew.sourceforge.net/>):** OpenGL extension wrangler
- **glfw (version 3.3.2, <https://www.glfw.org/>):** Windowmanger and input
- **glm (version 0.9.9.7, <https://glm.g-truc.net/0.9.9/index.html>):** Mathematics
- **stb\_image.h (version 2.25, [https://github.com/nothings/stb/blob/master/stb\\_image.h](https://github.com/nothings/stb/blob/master/stb_image.h)):** Image loading
- **Assimp (version 3.1.1, <https://www.assimp.org/>):** Model loading
- **Physx (version 4.1, <https://developer.nvidia.com/physx-sdk>):** *Physics simulation / collision detection, no longer used*
- **irrKlang (optional) (version 1.6.0 <https://www.ambiera.com/irrklang/>):** Audio output (music, speech, soundeffects (for example windsounds))