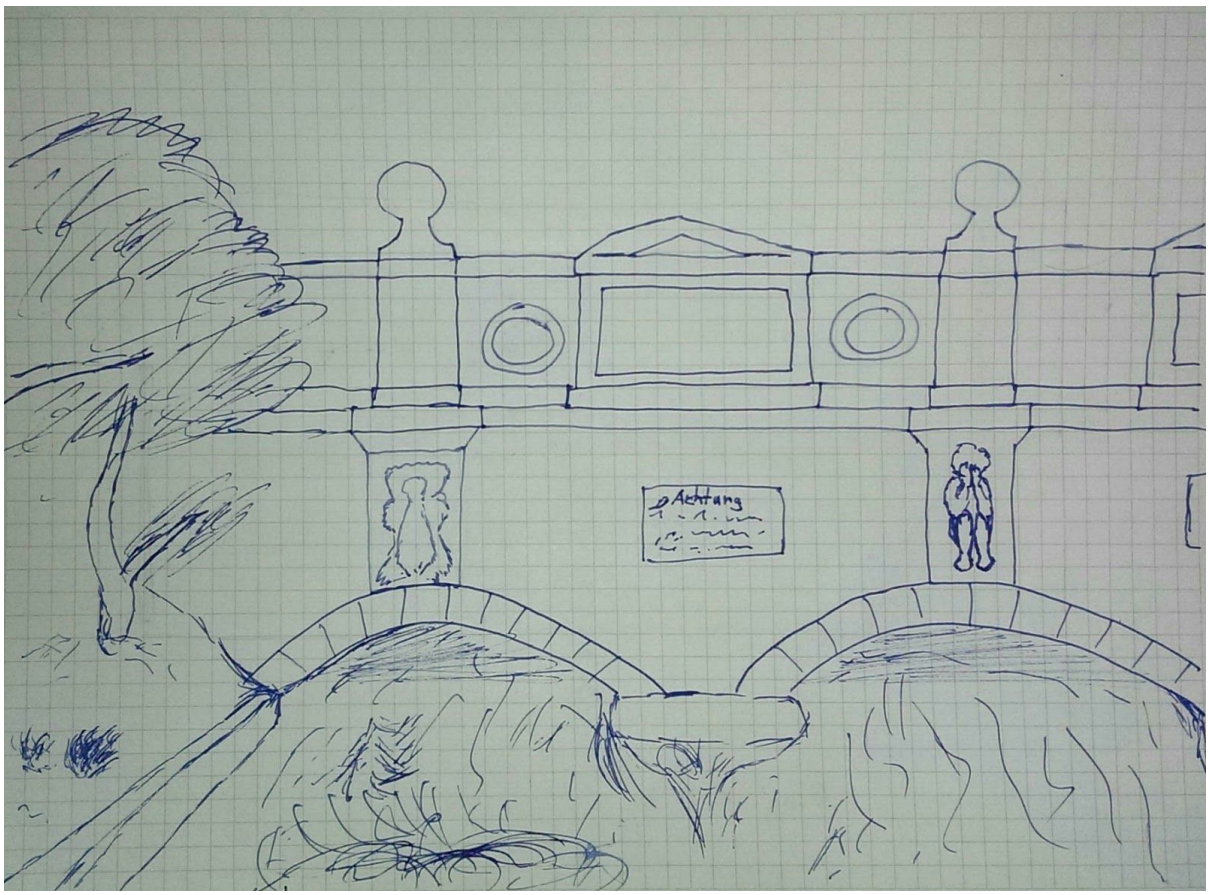
Game Idea Proposal

1. Game Description

The game “Waverider” (subject to change) will be a skill based reaction game with realistic fluid dynamics and rigidbody interaction. It supports the thematic requirement “Munich” with the “Eisbach”, a quite popular location for river surfing [https://commons.wikimedia.org/wiki/File:Eisbach_surfen_v1.ogv].
The standing wave is a unique place, so we decided to create a game as unique as the wave itself.

The idea of the game is to create a realistic surf simulation with similar difficulties like as in real surfing. The player has to stand as long as possible on the surfboard while the forces applied to the board by the river always forces the player to readapt its positioning.
The score depends on how long the player can stand on the board in the wave, maybe there are also some additional points for using difficult movements and tricks. We could also add a sort of a multiplayer-mode or let flow in some obstacles. This will give some additional challenge for skilled gamers.



In the multiplayer-mode two player could be playing on the same screen and trying to knock the other from the board without losing the balance.
In the second possibility the player has to evade the inflowing obstacles, without falling from the board.

A third person camera view will follow the surfer riding the wave while the player has to shift its point of mass on the board to not fall down. For this a special input device will be needed, which can track the position of a device via a gyroscope (Steam Controller, BeagleBoard) or the position of the user on a sensor board like the Wii Balance Board. A supporting GUI element will show the current pressure point by the water forces as a highlighted dot on the board the user has to reach to keep the board in place on the river. For this the user's center of mass projected on the board will be shown as a pulsating dot for clearer visibility and understanding of what has to be changed to keep standing on the board - or what the reason was to fall down into the river.

The base for the standing wave in the game will be one or more precomputed liquid simulations, running in an endless loop. This allows us to work with a high-resolution liquid simulation with low performance. Multiple simulation could be used to generate a more random behaviour. Additional details and interaction can then be computed during runtime by using particle effects and some simple modification of the precomputed data. For the data representation we decided to use an eulerian representation, using a SDF-grid. This allows us to compute really easy the buoyancy of the surfboard and gives us the possibility to render the fluid efficiently with a raytracer.
Apart of this we will focus strongly on the interaction between the surfboard and the water. This includes buoyancy, the flow forces and also collision detection, but also the rigidbody forces, like gravity and torque. The use of a SDF-grid as representation of the fluid makes it

easier to get the properties of the collision between the surfboard and the water. The goal will be to create a really realistic comportment of the surfboard on the water, which simulates the difficulties of surfing. Like for example if the velocity of the surfboard is too low, it will sink.
For finer visuals we also focus on a realistically oriented rendering of the scene. This combines the rasterizer based rendering pipeline for the surfer and the environment but also a raytracer which will render the water and additionally the bubbles of air under water, the spray on the water surface and the water drops in the air. With this combined approach we can generate visually stunning effects on the camera, assisting the focus of the simulation of the strong and fast river.

The surrounding will be composed of the Eisbach bridge, the riverbanks and some trees. All of them need to be modeled and textured, so the primary focus lies on the bridge and the surfer.
To combine all the technical parts in an efficient way, we will create our own engine from scratch. The Tsunami Game Engine will consist of the graphical factors of the rasterizer pipeline and a raytracer, while a custom wave simulation takes care of the river generation. Additional hierarchies have to be implemented to support different game objects with material properties. Also semi transparent GUI elements have to be included to play the game, besides an input controller.

2. Technical Achievement

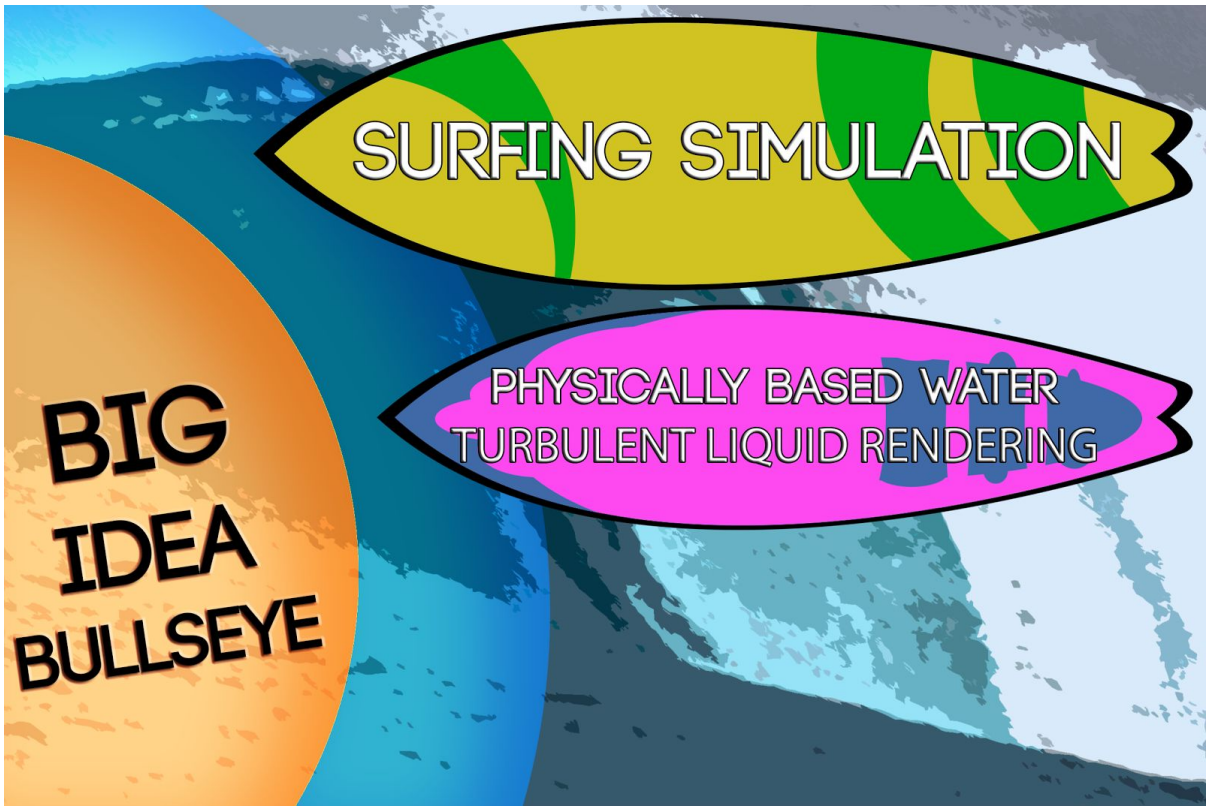
The core technical achievements of this project consist of the physically correct interaction of the wave and the surfboard with surfer, the custom rendering of the scene and the water, and the construction of a game engine combining these core elements, the Tsunami Engine. The Tsunami Game Engine is custom made, built for this game, but it can easily be extended to accommodate different genres of games.

At the heart of the technology there's a pure component-based architecture that lets users construct game entities in an agnostic manner, because of this feature programmers can create script classes to control the behaviour of each object and access the core layer of the engine to allocate and create resources directly from scripts.
Tsunami rendering engine is written in native *DirectX11* using a modern design based on APIs philosophies such as *Vulkan* and *Direct3D12* exploiting command buffers and pipeline objects design for a solid and structured render layer. Rendering logic follows a simple principle, materials don't need to know about the shading technique used to render them enabling us to register a newly created material to a render queue. This object will be then fetched at render time and submitted to the compliant pipeline. To avoid a huge number of context switches, Tsunami Engine uses a texture binding technique called *Megatexturing* introduced by *id Software* in *id Tech 6*. This technique is based on the idea that objects sharing the same pipeline or render pass will use the same huge texture containing all needed information for the material to render. Therefore, materials registered to a pipeline need only to know the index of their data in the mega texture avoiding a device context switch to bind a different resource.
Tsunami Engine also has compute functionalities accessible as volumetric rendering using GPU ray casting and post-processing chain. The first is a feature specifically tailored to render the wave for the game so it uses a fairly complex and very specific rendering pipeline designed to have a mixed rendering approach to the visualization of our game world. The latter is a simple collection of predefined and customizable series of compute shaders to realize image processing effects of the framebuffer prior to present. As the name suggests, image effects are executed in the order predefined by the user.
Last, but not less important is the GUI layer. At the time of writing this feature is not present in the engine but will let the user specify GUI materials and rendering info to blend alpha-valued images on the screen at desired position as overlay or text display.

For the physical part we will write also our own physics engine, which will provide some simple rigidbody physics and advanced liquid-rigidbody interactions.
We start by precomputing the wave as a liquid simulation using for example the FLIP method with a high resolution. The result will be imported in the engine represented as a SDF-grid. During runtime we can then extract the pre-simulated data and show it in a loop. Later on we could use also more pre-computed simulations which differ slightly in some properties, like for example in the inflow velocity and interpolate them to generate a less deterministic behaviour of the water.
Real-time effects and some random behaviour can then be added by using particle effects or some small modification of the SDF-grid. The particles can be generated in areas of high curvature on the surface, on collisions with solid objects or we could let flow them in randomly.
For the interaction between water and rigidbodies, like for example the surfboard, we use the advantages of SDF-grids. We can easily map the vertices of the surfboard to the SDF-grid and check if the values of the corresponding cell are water or not. Not only this, we get also the distance to surface. With this method we can calculate easily the buoyancy.
Finally for the flow forces we could use a static flow force which remains equals for the whole liquid, or we could also import additionally the flow velocity of the precomputed liquid simulation.

Our physics engine will support GPGPU using CUDA C. This allows us to accelerate the computation on the SDF-grid and will result hopefully in a good-looking and performant game.

3. “Big Idea” Bullseye



4. Development Schedule

See Attachment.

5. Assessment

This game will be all about reaction and observation. The player has to adapt the incoming water masses and continue to surf on the produced wave. The longer the player holds out while adding movement to his surfing, the higher the score will be. This game will appeal to a more mature or challenge seeking audience, who will try to beat the highscore of their friends.

The main strength of the game will be the physically accurate behaviour of the surfer with the wave. While maintaining balance on the board, the player will manipulate the river dynamics by visual recognizable effects, which will add an entertaining reward to this fun but challenging experience.