Lumen Force

Lou Kramer – Hubert Cao – Tobias Weiher – Manuel Dahnert

Computer Games Laboratory

Summer 2016

Contents

Game Proposal

Game Description - Technical Achievements - „Big Idea“ Bullseye - Development Schedule - Assessment

Game Description

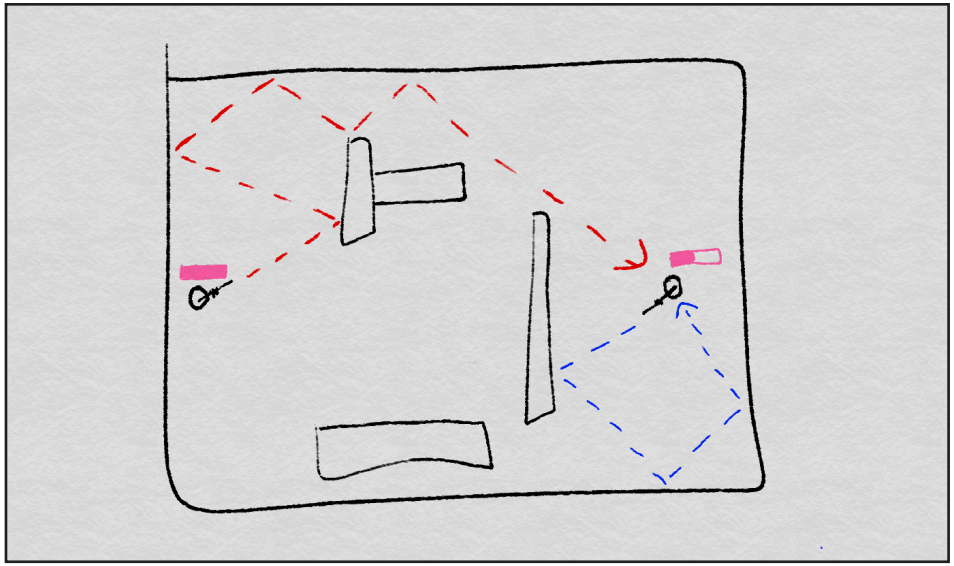
The game “Lumen Force” is a top-down shooter focused on arena levels. It implements the course theme “Arcade” via the core aspects of representative arcade games. These elements are the fast-paced action combined in an arena shooter with a top-down perspective supporting local player-versus-player (PvP) skirmishes.

Our game focuses in this course on the PvP fights with the goal of being the last player being alive, standing in the arena. For this we need to fulfill a promising situation for players to have fun and to show skills at the same time. The game takes place in a Sci-Fi setting. Thus, the players are equipped with futuristic weapons, shooting light projectiles with different radiant power and other light properties. This leads to different amount of damage being dealt on a hit player by one of the various projectiles. To increase the game depth each projectile can be reflected by every object in the scene. Therefore a covered player behind two obstacles can still be hit by a well targeted projectile on the neighboring walls to bounce off and reflect towards the enemy player.

However, they are not only reflected, they also follow the rules of light properties, meaning reflected projectiles lose in power, as part of it is absorbed in the obstacle. Depending on the obstacle’s material, an additional weaker reflecting projectile for diffuse surfaces or a transmitted projectile is spawned. The transmission follows the refraction rules by Snell’s Law. Further information on the light physics is listed under Technical Achievements.

Additionally, the absorbed energy can destroy the obstacles. This breaks the obstacle into smaller ones, which can hurt the player standing close to the moving fragment, though it also leads to cover on other positions. If the obstacles get too small, they vanish as dust. See for more information on destruction the Technical Achievements.

Because the number of projectiles in flight can reach a great amount, players do not get defeated by one hit. They have a health bar, which decrements depending on the energy level of the projectile hitting the player. For future updates on the game there might be some health regeneration by Power Ups, but these will not be implemented in the first versions.



Reflecting Projectiles, players can hit themselves (Friendly fire).

Also different weapons as an explosion grenade, pushing objects away, or vacuum grenade, pulling objects into the center of the grenade, are targeted for the final game.

Our game is a 3D game with a top-down perspective, effectively a 2.5D game. The models are some simple assets with basic animations for the necessary actions.

For our controls we decided to use the Twin-Stick controls.

The left stick of the controller changes the player position.

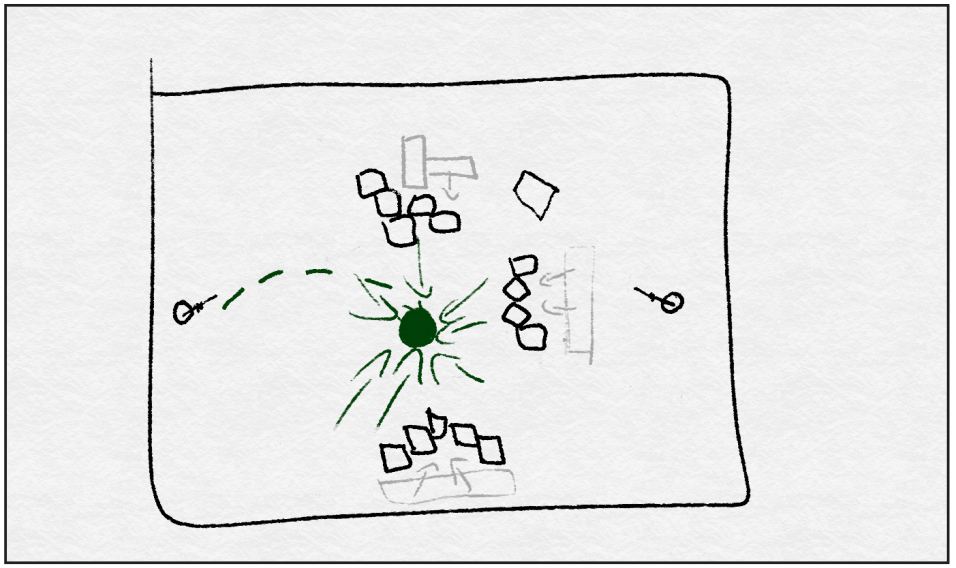
The right stick represents the orientation of the player, so a player can retreat while shooting in the opposite direction. This seemed to be a good solution for a fast-paced action shooter game without making the controls too complex.

The right trigger shoots light projectiles until the player has to reload or the weapon is overheated. The details have to be decided until a playable version can be tested, as these changes the gameplay feeling of the game depending on the strategic influence of either one.

The left trigger throws grenades, which can bounce off of walls and obstacles.

The arena map itself is composed of simple textures, as they should not distract the players from the action and the many projectiles in fight. Obstacles like deserted buildings or statues are randomly placed in the arena, so each fight is played differently.

For further versions of the game, there might be different gaming modes like Capture the Flag or a Co-Op wave scenario, where AI controlled enemies try to take down the players' base. The players then have to play together to hold out as long as possible. But they should be careful, as friendly projectiles can damage the teammates and oneself.



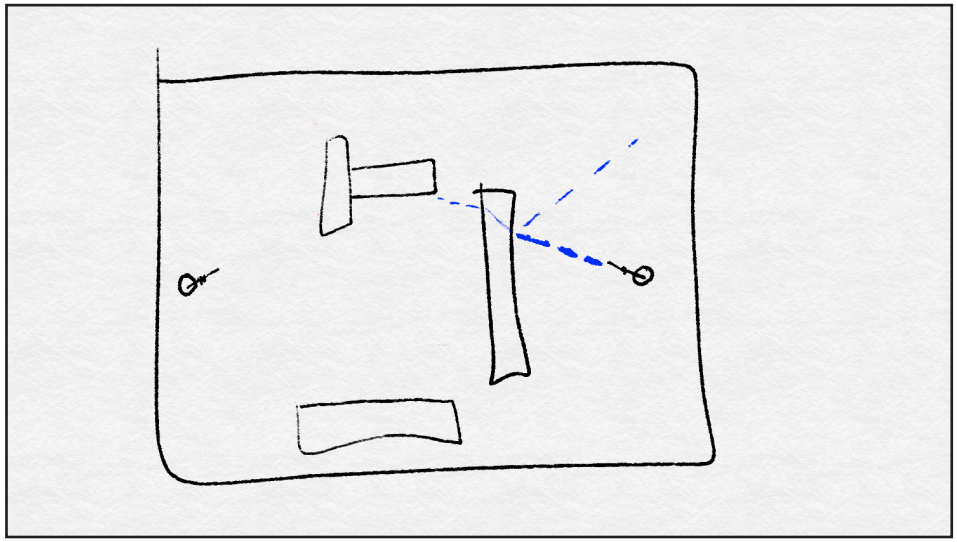
Vacuum grenades pull loose objects into the direction of implosion.

Technical Achievements

The game comprises two major technical achievements, the reflection of laser shots incorporating the properties of light and the destruction of obstacles resulting in smaller, separated sub-pieces.

Instead of having simple reflections of bullets, using the incoming force and direction for calculating the outgoing force and direction, the game uses laser weapons. Laser is “light amplification by stimulated emission of radiation” , thus considering light properties seems to suggest itself.

Once having this fact in mind, we can include the principles of transmission/ scattering, absorption and reflection of light rays, whose radiance also indicates the amount of damage a player receives being hit.



Transmitting and reflecting projectiles.

The effect of scattering is realized by applying Snell’s law, such that the refracted ray through the hit medium is computed, affecting only the voxels of the obstacle which intersects the ray. We also can define it as a local effect within the obstacle. In comparison, absorption acts globally on the whole obstacle. Thus, a specific amount of energy, using the imaginary part of the refractive index of the hit medium, gets absorbed and distributed among all voxels equally within the obstacle.

Reflections are composed of a specular part and a diffuse part. The specular reflection results in a laser ray of pretty high energy, depending on the absorption and transmission. If the amount of energy is higher than a certain threshold, a specular reflected laser ray can get reflected again by another object. However, diffuse reflections are vast in numbers but contain very low energy, which deals only little damage on the players and in case it hits another obstacle, it gets completely destroyed. This prevents an over-scaling of the amount of active rays travelling through the scene and interacting with the actors.

All effects on the laser projectile are scaled by the material properties and the initial values of the laser projectile itself. Thus, different laser shots and obstacles yield to various behaviors and results within the scene.

As already indicated above, the obstacles in the arena are composed of voxels, which contain information about the position within the obstacle and about the amount of received damage in a scale from 0 to 1, while 1 indicates no damage received and 0 indicates completely destroyed.

In this way, the destruction concept of the game is realized. Representing the objects with voxels, the extraction of subparts is computed by removing the respective voxels away from its original obstacle and building up a new one.

Getting back to the transmission example from above, all intersecting voxels with the scattered ray are computed and for each the received damage information is updated. Once a voxel gets completely destroyed, which is the case if the received damage information holds a value $> \varepsilon$, while $0 < \varepsilon \leq 1$, and if a connecting line of "broken" voxels from one surface to another is detected, the obstacle is cut. Since the line is restricted to lay on the xy-plane, because it is not possible to shoot up- or downwards, the orientation of the cutting plane is also restricted.

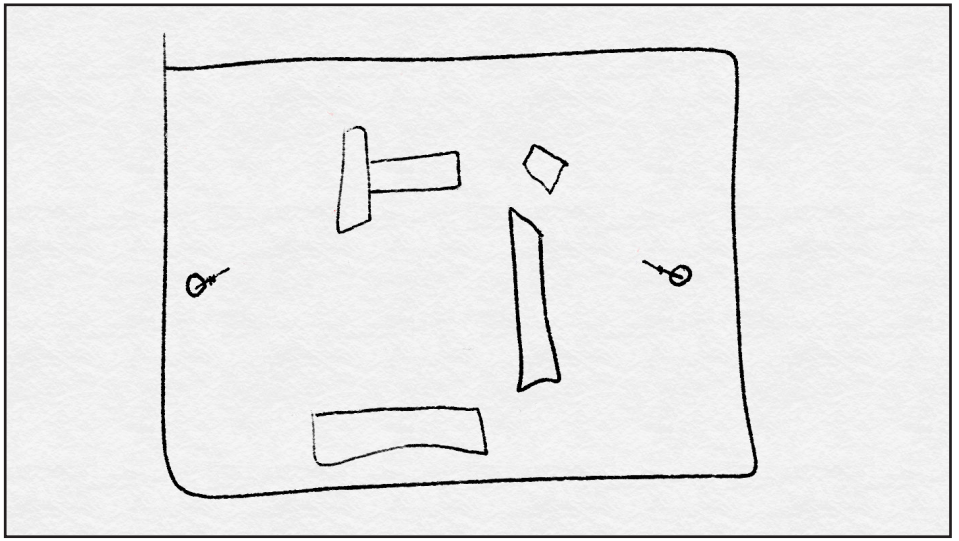
All cut voxels are deleted, causing forces pushing the two separated subparts of the obstacle away from each other. This results in two smaller obstacles at different positions than the former, bigger obstacle.

This concept might lead to very small obstacles, which have no real effect on the gameplay because of their size. To prevent this, a threshold is introduced and every obstacle smaller than a reference size is deleted, creating dust on their position to indicate a complete destroyed obstacle. Also, an obstacle which absorbs an amount of energy explodes completely without creating new, smaller sub-pieces, since all voxels received the same amount of damage.

Minor sub technical achievements are a random map generator and artificial intelligence for computer controlled enemies.

The random map generator shall fulfill certain constraints, such as not overloading the map with obstacles, creating corridors between them and in an even more advanced setting differentiate between different degrees of difficulty.

The field of AI can be explored in a wide manner, however the game focuses – if even included – on a very basic level of AI, such that the enemies do not get stuck at obstacles which refers to path finding and such that they shoot at the players.

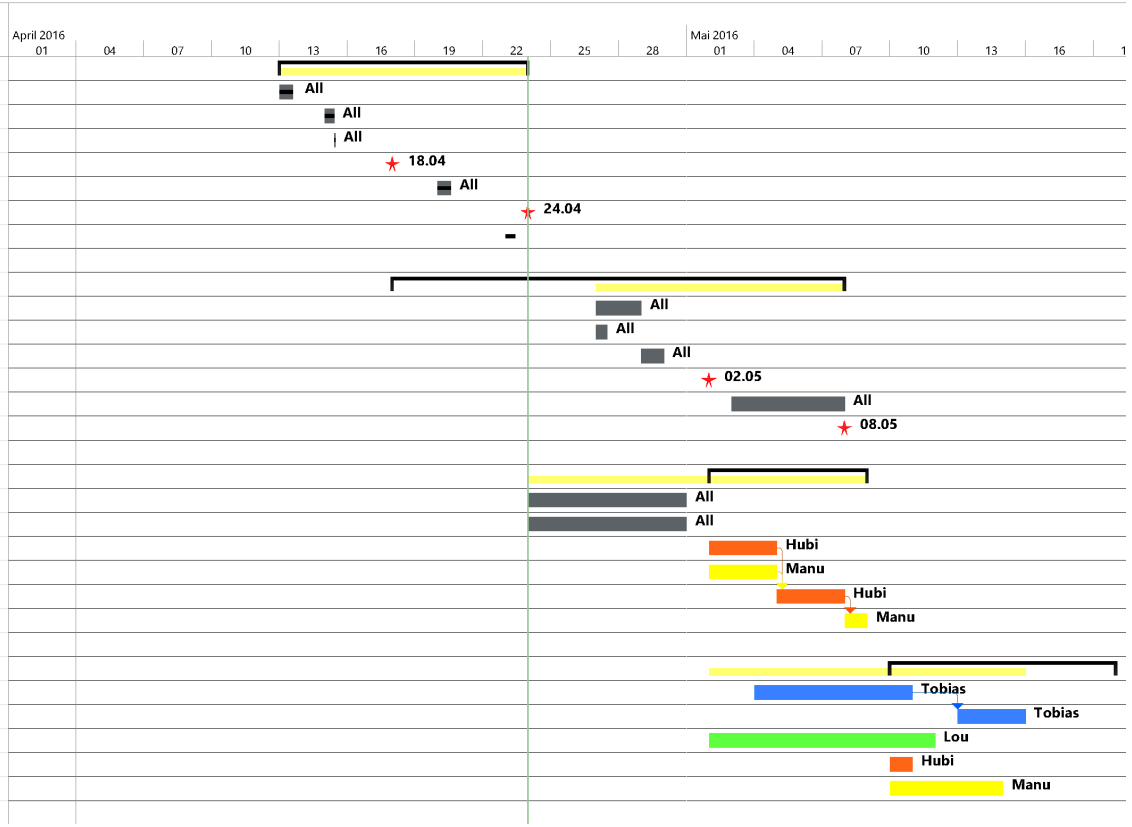


Obstacles can be destroyed in sub-pieces.

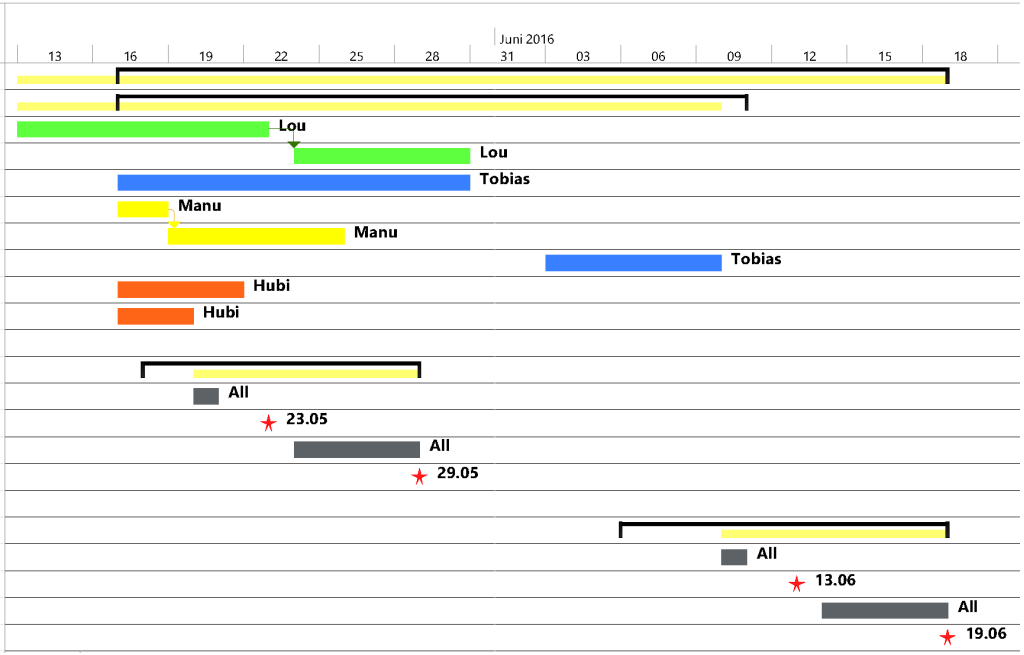
"Big Idea" Bullseye

Game Proposal » Developement Schedule

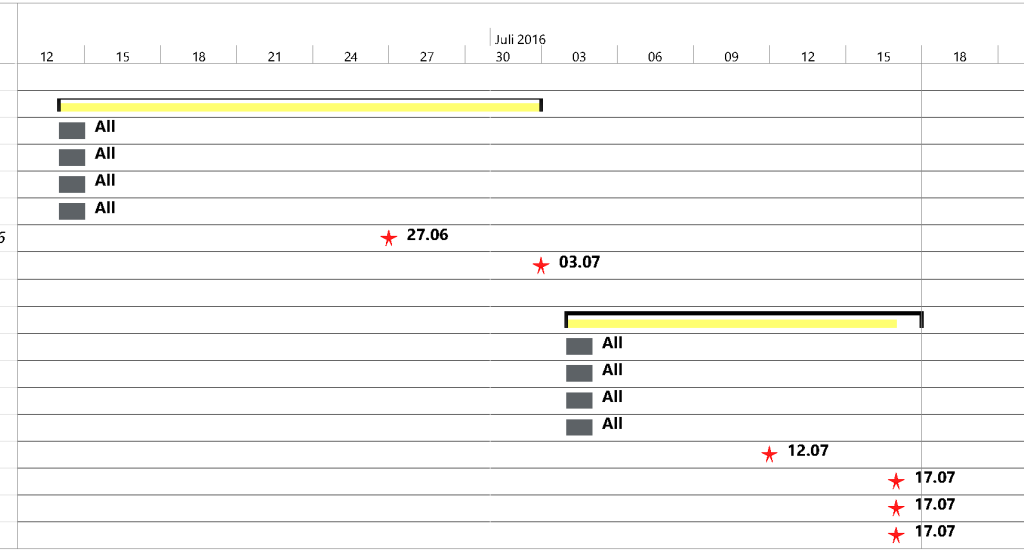
Nr.	Vorgangsname	Ist-Stunden	Geplante Stunden	Dauer	Anfang	Ende
1	**Game Proposal**			**11 Tage**	**Mit 13.04.16**	**Son 24.04.16**
2	Brainstorming	6	3	6 Std.	Mit 13.04.16	Mit 13.04.16
3	Finalize Game Idea	2	1	2 Std.	Fre 15.04.16	Fre 15.04.16
4	Prepare Game Idea Presentation	1	2	1 Std.	Fre 15.04.16	Fre 15.04.16
5	*Presentation: Game Idea*			*0 Tage*	*Mon 18.04.16*	*Mon 18.04.16*
6	Write Game Proposal Chapter	6	6	6 Std.	Mit 20.04.16	Mit 20.04.16
7	*Report: Game Proposal*			*0 Tage*	*Son 24.04.16*	*Son 24.04.16*
8	Write Critiques	2	2	2 Std.	Sam 23.04.16	Sam 23.04.16
9						
10	**Physical Prototype**			**20 Tage**	**Mon 18.04.16**	**Son 08.05.16**
11	Prototype Idea		4	2 Tage	Mit 27.04.16	Fre 29.04.16
12	Create Prototype		4	4 Std.	Mit 27.04.16	Mit 27.04.16
13	Prepare Prototype Presentation		4	1 Tag	Fre 29.04.16	Sam 30.04.16
14	*Presentation: Prototype*			*0 Tage*	*Mon 02.05.16*	*Mon 02.05.16*
15	Write Prototype Chapter		6	5 Tage	Die 03.05.16	Son 08.05.16
16	*Report: Prototype*			*0 Tage*	*Son 08.05.16*	*Son 08.05.16*
17						
18	**Functional Minimum**			**7 Tage**	**Mon 02.05.16**	**Mon 09.05.16**
19	Setup environment		24	7 Tage	Son 24.04.16	Son 01.05.16
20	Design Code structure		12	7 Tage	Son 24.04.16	Son 01.05.16
21	Controls		4	3 Tage	Mon 02.05.16	Don 05.05.16
22	Arena Map		4	3 Tage	Mon 02.05.16	Don 05.05.16
23	Local Multiplayer		8	3 Tage	Don 05.05.16	Son 08.05.16
24	Player Distinction		2	1 Tag	Son 08.05.16	Mon 09.05.16
25						
26	**Low Target**			**10 Tage**	**Die 10.05.16**	**Fre 20.05.16**
27	Projectile Reflections		15	7 Tage	Mit 04.05.16	Mit 11.05.16
28	Weapon Overheat		2	3 Tage	Fre 13.05.16	Mon 16.05.16
29	Voxelized Map Assets (Obstacles)		30	10 Tage	Mon 02.05.16	Don 12.05.16
30	Health Bar		2	1 Tag	Die 10.05.16	Mit 11.05.16
31	Basic Assets (Characters)		6	5 Tage	Die 10.05.16	Son 15.05.16
32						



Nr.	Vorgangsname	Ist-Stunden	Geplante Stunden	Dauer	Anfang	Ende
33	**Desirable Target**			**33 Tage**	**Die 17.05.16**	**Son 19.06.16**
34	**Features**			**25 Tage**	**Die 17.05.16**	**Sam 11.06.16**
35	2D Destructions		20	10 Tage	Fre 13.05.16	Mon 23.05.16
36	2,5D Destructions		10	7 Tage	Die 24.05.16	Die 31.05.16
37	Advanced Reflections		30	14 Tage		
38	Menu / Settings		5	2 Tage	Die 17.05.16	Don 19.05.16
39	Sound / SFX		8	7 Tage	Don 19.05.16	Don 26.05.16
40	Random Map Generation		15	7 Tage	Fre 03.06.16	Fre 10.06.16
41	Grenade		8	5 Tage		
42	Different Weapons		4	3 Tage		
43						
44	**Interim State**			**11 Tage**	**Mit 18.05.16**	**Son 29.05.16**
45	Prepare Interim Presentation		3	1 Tag	Fre 20.05.16	Sam 21.05.16
46	*Presentation: Interim Demo*			*0 Tage*	*Mon 23.05.16*	*Mon 23.05.16*
47	Write Interim Report		8	5 Tage	Die 24.05.16	Son 29.05.16
48	*Report: Interim Results*			*0 Tage*	*Son 29.05.16*	*Son 29.05.16*
49						
50	**Alpha State**			**13 Tage**	**Mon 06.06.16**	**Son 19.06.16**
51	Prepare Alpha Presentation		3	1 Tag	Fre 10.06.16	Sam 11.06.16
52	*Presentation: Alpha Demo*			*0 Tage*	*Mon 13.06.16*	*Mon 13.06.16*
53	Write Alpha Report		8	5 Tage	Die 14.06.16	Son 19.06.16
54	*Report: Alpha State*			*0 Tage*	*Son 19.06.16*	*Son 19.06.16*



Nr.	Vorgangsname	Ist-Stunden	Geplante Stunden	Dauer	Anfang	Ende
70						
71	**Playtesting**			**19 Tage**	**Die 14.06.16**	**Son 03.07.16**
72	Conduct Play Testing					
73	Play Testing results					
74	Polish Game					
75	Prepare Playtesting Presentation					
76	*Presentation: Playtesting Results*			*0 Tage*	*Mon 27.06.16*	*Mon 27.06.16*
77	*Report: Playtesting Chapter*			*0 Tage*	*Son 03.07.16*	*Son 03.07.16*
78						
79	**Final Presentation**			**13,38 Tag**	**Mon 04.07.16**	**Son 17.07.16**
80	Prepare Final Presentation					
81	Create Video					
82	Write Final Documentation					
83	Ship Final Game					
84	*Presentation: Final Presentation*			*0 Tage*	*Die 12.07.16*	*Die 12.07.16*
85	*Report: Consulsion Chapter*			*0 Tage*	*Son 17.07.16*	*Son 17.07.16*
86	*Report: Video*			*0 Tage*	*Son 17.07.16*	*Son 17.07.16*
87	*Report: Final Version*			*0 Tage*	*Son 17.07.16*	*Son 17.07.16*



Assessment

With the proposed game as very action packed and fast paced, quick reactions and fast decision making are required to play the game. These properties and the Sci-Fi setting shall appeal especially to a younger audience. The coolest feature is the light property incorporated by the projectiles, such as reflection, transmission and absorption.

Although it is an action filled arena shooter, the players have to think outside the box. While dodging other projectiles, they have to quickly come up with a strategy to hit their opponents using their own reflecting projectiles and taking advantage of destructible obstacles. Herein lies the main strength of the game. A weakness could be the lack of a game element which provides long-term motivation.

The smart combination of the action and strategy elements shapes the success of the game. The strategy component must not slow down the pace of game, but on the other hand it has to be relevant and contribute to gameplay.

Physical Prototype

Prototype Idea - Experience - Design Revisions

Prototype Idea

With a physical prototype of the game concept, we simulated the core elements and tested the game dynamics under the aspects of fun, strategic play style and game flow. A physical prototype has the advantage of a fast realization of concept changes and a quick investigation of their impact to the game.

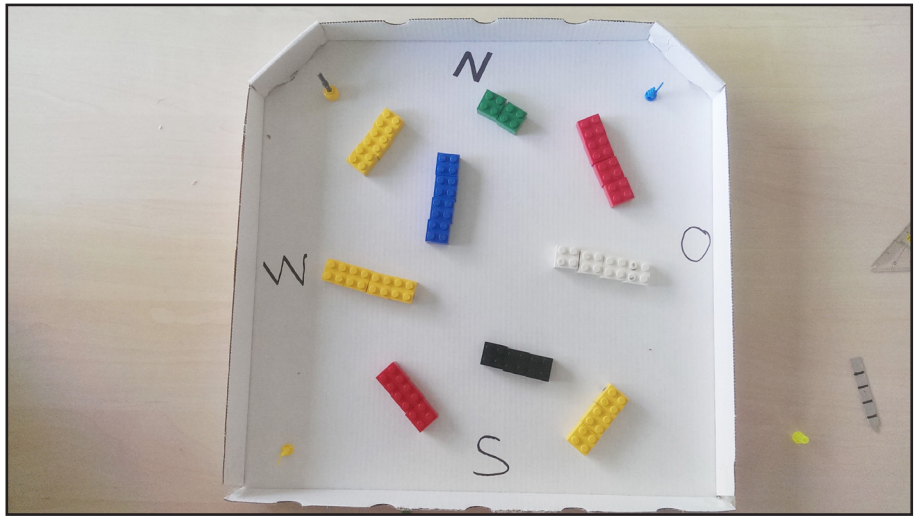
Game Objects

The prototype is composed of an arena, players and a game master.

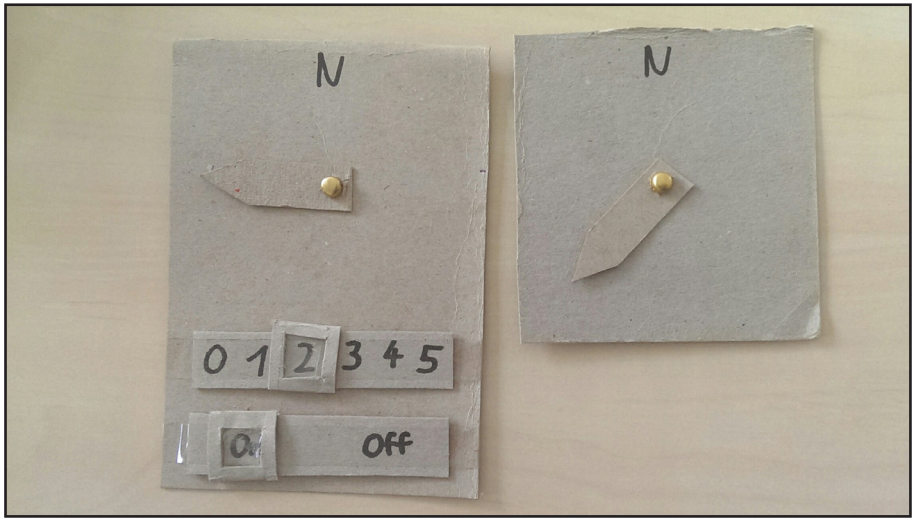
Arena The arena is a planar quadratic shaped platform with walls at the boundaries and rectangular obstacles composed of several cubes. While the walls are fixed and indestructible, the obstacles can take damage, get destroyed and split up in two pieces. The players spawn in the corners of the arena.

Player Controller The players can move in any direction, which means they have 360° of freedom. This direction is indicated by an arrow of the player controller card in an absolute manner. The moving distance ranges from 0 to 5 centimeters per round, specified by a slider.

Besides moving, the players can also shoot in any direction, defined by a second arrow.



A carton serves as the arena platform and the four directions of the compass (wind rose) are marked for reference. Small figures (e.g. Risiko soldiers) represent the players. LEGO elements are used for the obstacles.

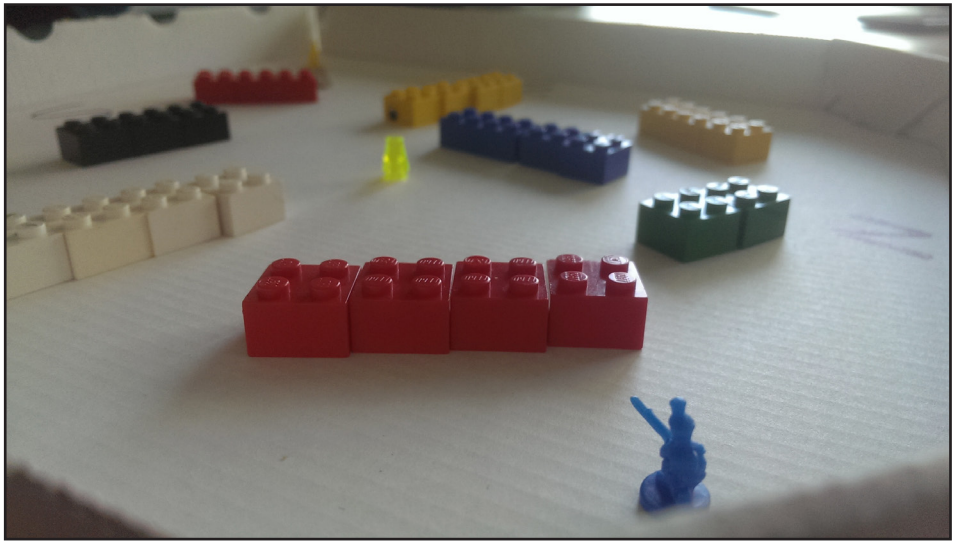


On the left card, the absolute movement angle and the movement distance are specified. Also whether the player wants to shoot or not. On the right card, the absolute shooting direction is stated.

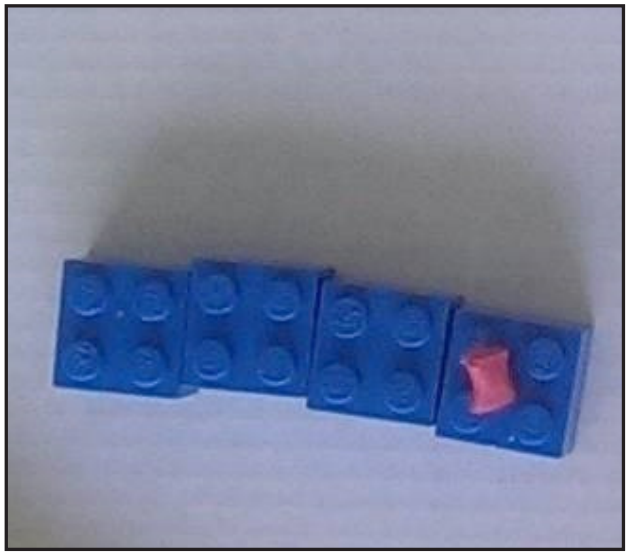
Player Stats The players have a health bar, starting with 5 points. If a player loses all hit points, he or she dies and will not respawn.
Every round, the players are able to shoot one projectile with an energy level regarding to their current weapon heat. Every time the players shoot, their respective weapon heat increases about 1, if they do not shoot, it decreases about 1. If the weapon heat reaches the highest heat level, which is in the prototype level 6, the player has to wait one round before shooting again because the weapon is overheated.

Projectiles The projectiles, shot by the players, are classified in five energy levels, ranging from the highest, blue, to the lowest, red. If a player is hit, the player receives damage regarding to the energy level of the projectile, while blue deals 5 damage and red 1. See also table 1.
If the projectile hits an obstacle, it gets reflected and one energy unit is absorbed, so e.g. a blue projectile turns into a green projectile. Also, the cube of the obstacle, which was hit, receives one damage unit.
If the projectile hits a wall, it also gets reflected and loses one energy level.

Obstacles Obstacles are composed of several cubes, each of them can absorb one energy unit. If a cube receives two energy units, it gets destroyed and the obstacle is split and slightly moved to show the destruction impact.



The different colors of the elements do not have any impact of the properties of the obstacle.



Damaged obstacle cubes are symbolized by putting the absorbed projectile on top.

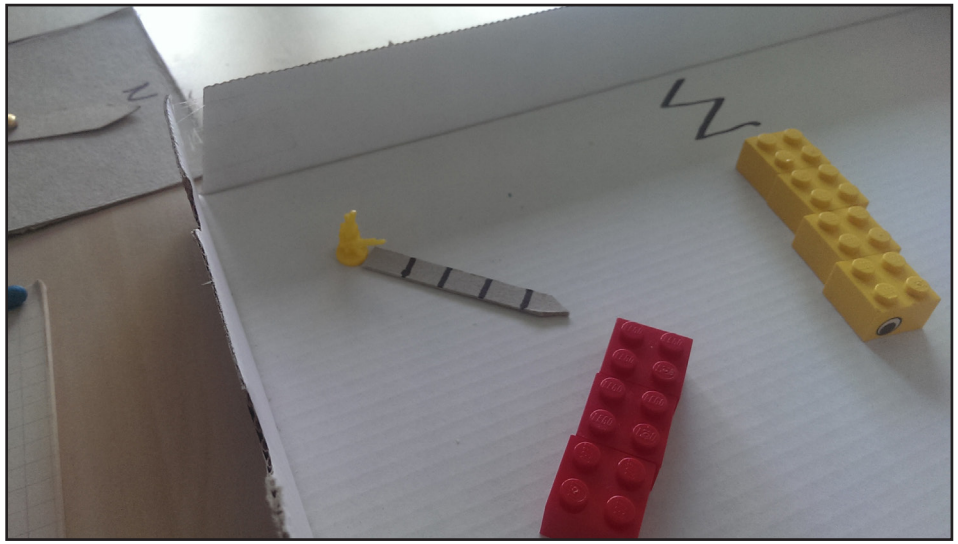
Game Structure

Since simulating multiple real time elements is difficult in a physical environment, especially the projectiles, the game play is round based. Each round is composed of two phases, the planning phase and the execution phase.

Planning Phase All players plan their next actions on their controllers at the same time, but hidden from each other. They define the absolute movement direction and the movement distance, state if they want to shoot and if yes, define also the shooting direction.

Execution Phase Once every player has finished the planning phase, everyone reveals their controller cards and the game master executes the actions in a sequential order, but ensuring that the parallel character maintains.

First, every player is moved in the respective moving direction by the desired moving distance. Then, the projectiles, which are already in the game, are moved by 10 centimeters, while considering reflections, absorptions and obstacle destructions. After this, the game master places the new spawned projectiles 5 centimeters away from the shooting player in the respective shooting direction. The energy level of the projectile is set according to the current weapon heat of the player. At the end, the HP and weapon heat of the players are updated.



The players are moved by the Game Master according to the distance on the player card.

Blue	Green	Yellow	Orange	Red	overheated
5	4	3	2	1	0

The table shows the energy levels of the projectiles. If the weapon of the player is overheated, the player cannot shoot and has to wait one round.

Experience

The game play of the prototype turned out to be very strategic, since it required the overview of all active projectiles and the 360° freedom of both, the movement and the shooting direction. Also, the destruction of obstacles played a key role especially in the later game, when they got sparser and sparser.

However, even though the strategic element of the game was investigated by the prototype quite well, the fast game play style got a bit lost due to the game structure composed in two phases. To enforce shorter games with more action involved, sudden death, lesser hit points or adjustments of the weapon heat concept to support more active projectiles can be introduced. However, the more projectiles were active the more difficult it was for the game master to keep track of everything.

Design Revisions

During the creation of the prototype, several core elements were revised and further worked out. One of the main key elements is the weapon heat concept, which we discussed and reworked.

Also, the destruction behavior of the obstacles was clarified. Absorption now affects only the hit cube of the obstacle rather than the whole object.

Additionally, the projectile spectrum – the five energy levels – were defined according to the law of physics, in which blue light has a higher energy level than red light.

Possible extensions

The prototype creation also revealed several new game elements, which may serve for extensions of the game, but was out of scope for the prototype itself. We introduced the concept of having a power up, which allows the player to shot a projectile of white light, which e.g. kills a hit player instantly or splits up in 5 projectiles, one of each color at obstacle hit.

Furthermore, several obstacle types were developed, such as glass, bricks, sandstone, and wood, all of them having different behavior in terms of absorption and refraction, which was excluded from the prototype concept due to complexity issues.

Interims Release

Functional Minimum - Low Target - Desirable Target - Design Revisions - Challenges

Functional Minimum

General

Project planning During the game proposal phase, we were using Microsoft Project (out of curiosity) to create the development schedule. At this point Microsoft Project provided extensive features and functionality to precisely plan the project beforehand. However, it turned out, that firstly each team member needed the program to submit his or her progress. Secondly the extensive feature set was just an overload for the purpose of this project.
Due to these reasons, we created a separate page on the course Wiki and copied the five layers in form of simple tables. This Wiki page gives each team member the possibility to easily keep track of the project status and also to easily submit his or her contribution. It also serves as a persistent place for bug reports and general notes.
For the milestone deliveries the progress and working hours from the Wiki pages are copied back to the Microsoft Project sheet, since the program can visualize the progress in a quite informative manner.

Project Environment Already at the beginning of the course, we decided, that we are going to utilize the Unreal Engine 4. The latest version at the start of development was 4.11. The engine is written in C++, but also provides a visual scripting system, called "Blueprints". However, we want to use the latter one only where it is absolutely necessary and remain mostly with C++, which allows us to further improve our programming skills.
The main target platform is Windows, whereas it is also possible with Unreal Engine to deploy for Linux systems.
Further, we use Visual Studio 15, which is required since Unreal Engine 4.11, and control our progress with a Git repository on GitLab. Our Git follows no specific workflow, such as Feature branches or Gitflow, since merging the binary asset files of Unreal is a real pain and also quite error-prone.

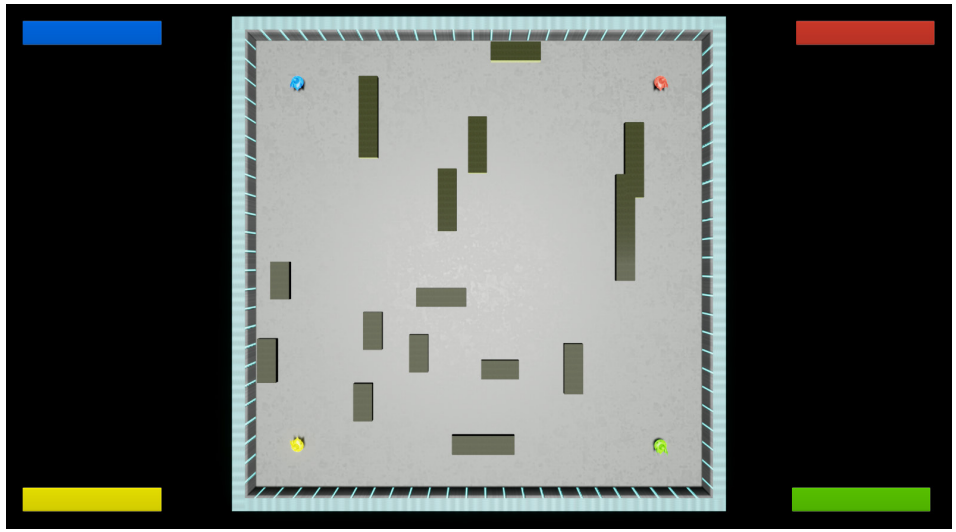
Gameplay

Controls Unreal Engine comes with a twin-stick control scheme, but it is very basic and insufficient for our purposes, so several adjustments had to be made. First of all, firing has to be controlled by means of a fire button; it is assigned to the right trigger of an Xbox controller. Secondly the player model has to face the direction of the right analog stick, which controls the shooting direction. Ultimately, if the right analog stick is not pressed, the player model faces the movement direction again. Regarding this point, we are not sure whether the orientation of the player model should rotate back to the movement direction or stay at the shooting direction. This has to be evaluated in the playtesting section and adapted accordingly.

Local Multiplayer Unreal Engine's native local multiplayer is implemented as a split screen multiplayer with a camera per player. Unfortunately, it is useless for our proposition since we have an arena shooter with one static camera for all players. Firstly, a camera which captures the whole arena has to be added. To replace the player camera, it has to be removed from the player model. Additionally, each player's view target has to be set to the arena camera. To ensure each player spawns when the game starts, custom spawn points with clever spawn logic have to be implemented, so players can only spawn at a point if no other player has spawned there before. Otherwise there will be conflicts and some players will not be added.
In order to distinguish each player from another, each player has a unique ID, which determines the color of the player model.

Obstacles and Arena The obstacles are composed of several cubes, which are generated with a procedural mesh generator. Therefore, the vertex buffer and index buffer are explicitly defined with a global fixed voxel size. To assure correct UV-mapping, 24 vertices are stored instead of only 8. With those generated cubes – also referred to as voxels in this report -, custom obstacles can be built. At the moment, the obstacle generator provides a function for building rectangles of variable size, but extensions for other shapes can be made.

To be consistent with the obstacles, the arena itself is generated with a procedural mesh generator. The size of the arena is variable, and the arena boundaries can also be specified in terms of thickness and depth. The camera depends on the arena size to show the whole arena.



Screenshot of the Interims state.

Projectiles Projectiles are currently set up by a capsule mesh and its own collision behavior for the world objects. Each projectile is shot by the player depending on his or her weapon heat and energy level. As of now, the projectiles only have one color for testing the changeability of the material properties. In later builds the color of each projectile will show its energy level from blue (high) to red (low).

If a projectile hits a player, all of its energy is converted to the damage inflicted on the hit player. Other hit obstacles either reflect or refract depending on their refractive index, used for the reflectance coefficient calculated by Schlick's approximation. This gives the percentage of reflectance in contrast to the refraction of certain materials and therefore the energy level of newly created reflected projectiles and the refracted one. The refracted projectile changes its direction by Snell's law, which also takes the refractive index of the two materials on the planar surface into account. The border walls only reflect and reduce the projectile's energy, so no projectile can leave the arena.

For the interims build, the projectile just deals damage to the players and currently not to the voxels/obstacles, as refractive indices are not yet integrated, but the functionality is already given.

Low Target

Gameplay

- The health bars are colored accordingly to the players and represent the health of the respective player.
- If a projectile hits a player, it applies damage to the player according to his or her current energy level.
- If the health of a player reaches zero, he or she dies. But since the view targets are bound to the players, simply destroying the player object on death will mess with the camera view. So if a player dies his or her controls are disabled and the model is removed.

Projectiles The current projectiles are not using the correct collision normal for the obstacles, which results in rather strange reflecting directions. For the interim presentation we fell back on a previous build, which used a debugging reflection direction for testing the collision, which simply multiplied the direction by -1, resulting in going back where the projectile came from. Reflections on the border walls are offering the correct normals and intended reflection behaviours. The previous build had a problem with continuous collision detection, which might lead to projectiles ignoring obstacles because of their high speed, and also projectiles being stuck inside an obstacle, as it has registered another collision frame by frame. These problems are already fixed by using the physics system of unreal, which tracks the path of projectiles and therefore checking for collisions in between two frames.
Also the projectiles should have a border color to show from which player the projectile was being shot. This is not yet really visible because of the speed of projectiles, so this might need some changes or a complete removal to support better understanding and clarity of projectiles in flight.

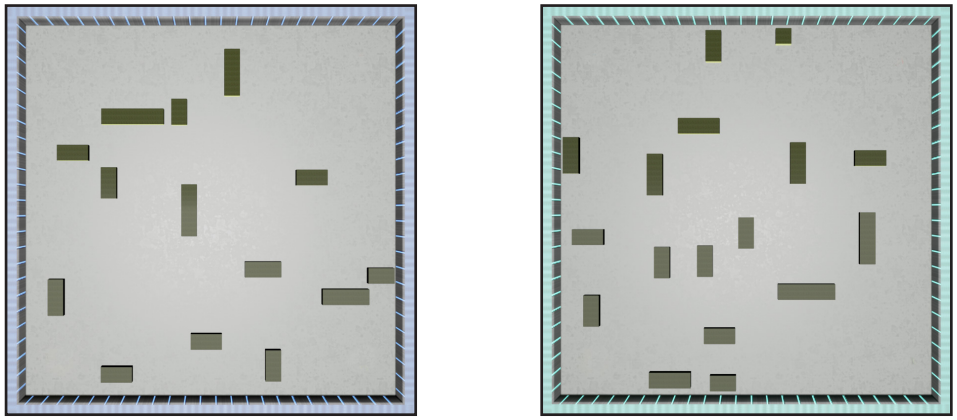
Desirable Target

Gameplay The menu at the interim state is purely a basic menu with the functionality of playing and quitting the game. However, it provides a starting point for a more advanced menu, which will be completed in the next development phase.

Obstacles To support destructions, a destruction handler is created. It is able to delete all destroyed voxels of an obstacle and to find separated sup-parts of voxels, of which a new obstacle can be generated. It starts at a random voxel and traverses in a 'first in first out' – queue manner all neighboring voxels, collecting them in a separated array. If all neighbors are visited and there are still unvisited voxels left in the obstacle, then a sub-group was found and a new obstacle needs to be created. This makes it possible to apply physics to the obstacles in a later state of game development, even after they got destroyed and separated in several pieces. Also, obstacles with a variable height in the z-component, which is not included at the interim report, can be added, since the search algorithm works independently of the axes because it calculates the distances between the center of voxels.
However, in the interims state destructions are theoretically possible, but not included within the game play, since the projectiles do not deal any damage to the obstacles yet.

Random Arena Generator Random generated arena layouts require the player to adapt to the different obstacle placements in order to fully utilize the light properties of the projectiles. This therefore leads to a higher game depth.

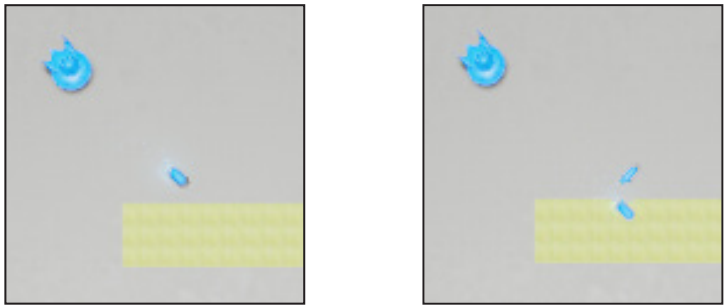
The Random Arena Generator divides the available arena size into N logical axis-aligned cells of random, but limited size. Within each cell one obstacle with random length is placed. Currently, the arena generator limits the obstacles to be either a horizontal or a vertical arrangement of voxels with a defined width.



Different layouts of obstacles generated by the random map generator.

Projectiles The advanced reflection for the desirable target includes the usage of refraction and applying damage to crossed voxels inside an obstacle, changing direction of the path by Snell's Law and changing energy level by Schlick's approximation for receiving the reflectance percentage and the remaining refraction energy. What we were missing for the interims build was the change in direction, as obstacles did not have the refractive index yet, and the change of color of each projectile depending on their energy level.

Projectiles also might need a trail of light following their path to create clarity for the players as they understand the direction of incoming projectiles. As Unreal Engine's particle system is quite complicated, we are currently using a simple, hardly noticeable particle system, which we will change in later builds.



A projectile splits up itself, get reflected and refracted.

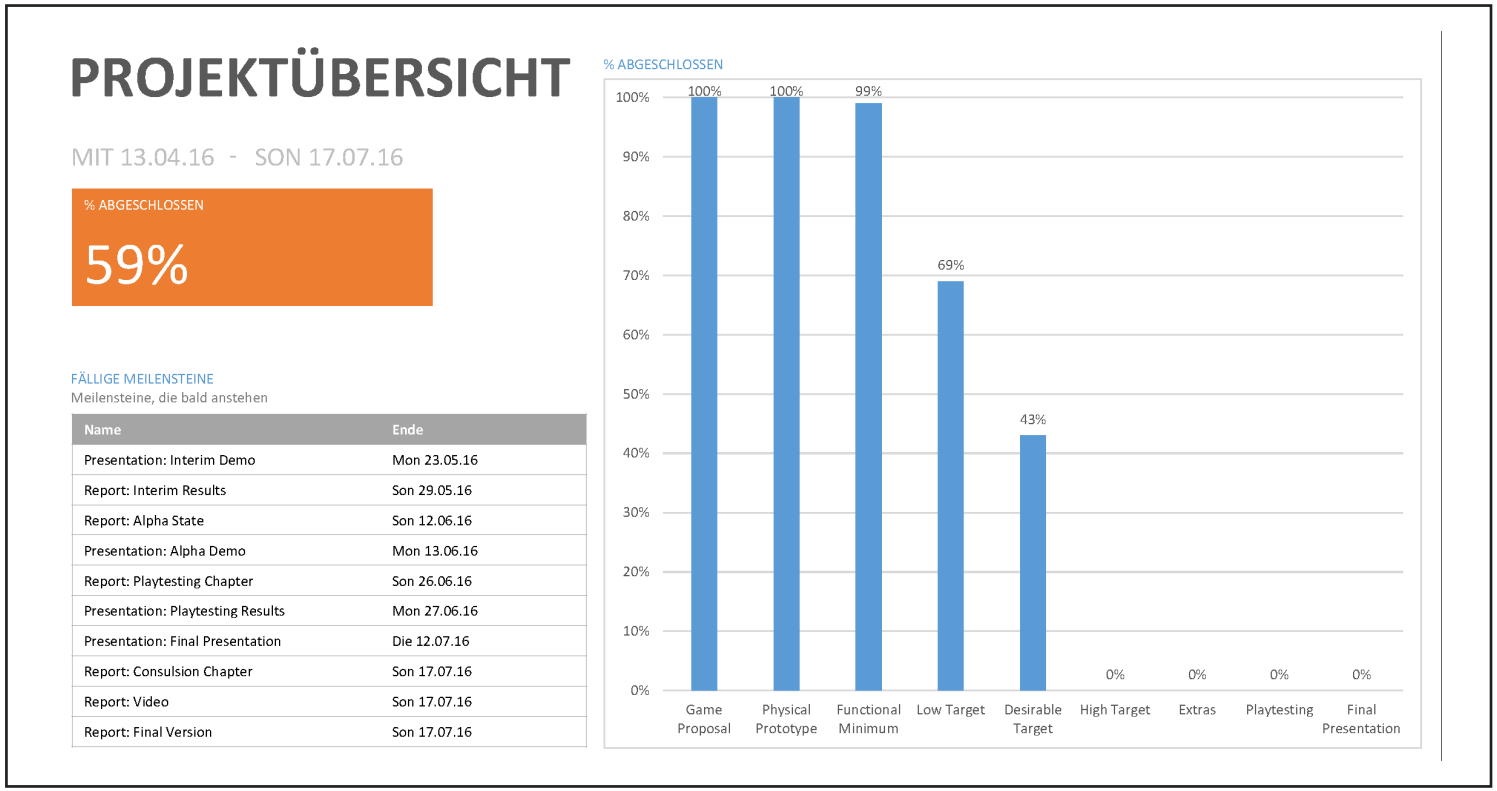
Design revisions

While most game elements currently only provide basic functionality, we did not change any aspect of our initial design decisions.

Challenges

Using an existing engine allows us to focus on the key elements of the game. However, it also requires careful studying of the game engine's architecture to prevent pitfalls. Additionally, we are in some cases limited by the engine's capabilities and design decisions.

The visual scripting system of Unreal Engine provides a fast way to add functionality without any programming. But since we are familiar with programming C++, we do not want to use Blueprints. However, there are some things, which can only be done with Blueprints.



PROJEKTÜBERSICHT

MIT 13.04.16 - SON 17.07.16

% ABGESCHLOSSEN

59%

FÄLLIGE MEILENSTEINE

Meilensteine, die bald anstehen

Name	Ende
Presentation: Interim Demo	Mon 23.05.16
Report: Interim Results	Son 29.05.16
Report: Alpha State	Son 12.06.16
Presentation: Alpha Demo	Mon 13.06.16
Report: Playtesting Chapter	Son 26.06.16
Presentation: Playtesting Results	Mon 27.06.16
Presentation: Final Presentation	Die 12.07.16
Report: Consulsion Chapter	Son 17.07.16
Report: Video	Son 17.07.16
Report: Final Version	Son 17.07.16

Overview of each layer and their progresses.

Alpha Release

Low Target - Desirable Target - High Target - Design Revisions - Challenges

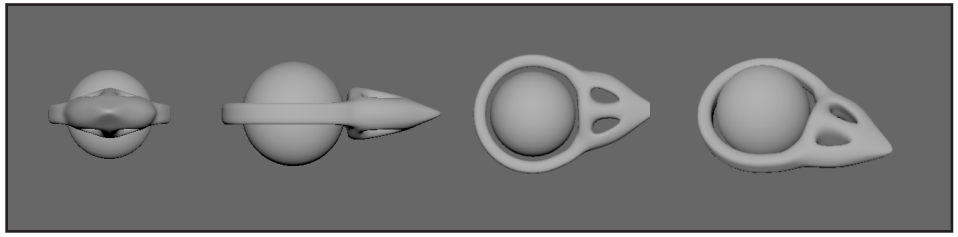
Low Target

Weapon Heat To ensure a certain amount of strategy and avoid overloading the arena with projectiles, we needed a game mechanic to prevent the player from continuous firing. Since a bullet capacity/reloading system does not really make sense in our light approach, we have decided to implement a weapon overheat system: Each time the player fires a projectile, his or her weapon overheat increases. Once it reaches the limit the weapon is overheated, the player is not able to keep shooting and has to wait until his or her weapon has cooled down. Additionally, the higher the overheat of a weapon is, the lower energy the projectile it fires has. To balance out the drawbacks of low energy projectiles, they produce smaller overheat than high energy projectiles. So the weapon overheat does not increase in a linear manner, but somewhat logarithmically. It looks as follows:

Current Weapon Overheat	Increase
0% - 5%	+30%
5% - 15%	+25%
15% - 30%	+20%
30% - 60%	+15%
60% - 100%	+10%

Advanced Projectiles: Colors The difficulty in applying the different colors regarding the projectile's energy level was to find an appropriate transition function between the colors blue over green to red. This was achieved by using the HSV colors, where the saturation and the value (brightness) both were kept at 1.0f, whereas the hue ranged from 240 for blue to 0 for red (120 is green), depending on the current energy level. This HSV color is then transformed into linear RGB values to change the final color. The Unreal Engine dependent property is to reapply the material to the static mesh to see the change in color, as UE cannot have runtime changeable materials directly.

Own Player Model We replaced the player model, which was given by Unreal, by a basic, but own 3D model. The model was inspired by a olling robot, which consists of a spherical body. The spherical body ensures the desired movement in arbitrary direction. An additional ring around the body represents the gun. This gun forms an arrow at one side to clearly indicate the shooting direction. The gun ring is detached from the body and simple floats in the air. This again ensures the 360° shooting direction.



Different views on the new player model.

Advanced Obstacles: Multiple materials To increase the variety of obstacles for enhancing strategic gameplay, multiple obstacle types were added with different refraction and absorption coefficients. The coefficients are based on the materials gold, glass, silicon, germanium and crystal/ diamond. The decision for those types are based on the availability of the refraction and absorption coefficients, which are e.g. not really defined for organic materials such as wood, but very well investigated for inorganic materials such as different types of plastic and metals. Another criterion is the visible distinction between the materials, so the player is able to determine the different obstacle types easily.
The types are implemented using inheritance. Therefore, a basic voxel class is created and then the specific material coefficients and textures are defined in the child classes.

Desirable Target

Obstacle destruction: Coupling with projectiles The obstacle destruction is managed by a global destruction handler. Since there is no master class which has an overview over all actors in the game, the destruction handler has to be stored in the projectiles, which are calling the test function after they damaged an obstacle. To ensure that not every projectile has its own destruction handler, but there is one global for every projectile in the scene, a singleton container is used. Every time a projectile is shot, the constructor checks if already a global destruction handler is available and if yes, sets the pointer within the projectile class. However, the first time a projectile is created, no global destruction handler is available, so one will be created.

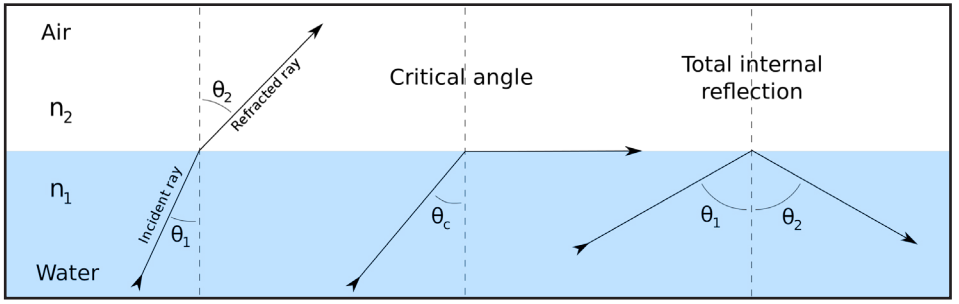
Advanced Reflections For advanced reflections, we needed to implement physical properties of light-matter-interactions. In reality, these can become quite difficult for certain materials, as they use the polarity of light to get the final refraction and reflection directions and intensities. For our real time application, we needed something less detailed and more focused on fast results. Therefore to receive the reflection coefficient (amount of reflection in contrast to refraction), we used Schlick's approximation for the contribution of the Fresnel factor in the specular reflection of light. This computes the relation of the reflection coefficient parallel to the normal plus the amount of difference with the cosine between incident light and normal. For this calculation, we also need refractive indices for the two materials, one being the air and set to 1.0f, the other one depending on the hit material.

Also for the light being refracted into the material, we need to change the direction. This is done by applying Snell's Law with the formula:

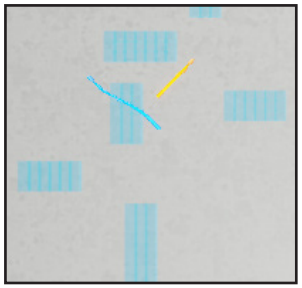
$$n_1 * \sin\theta_1 = n_2 * \sin\theta_2$$

It changes the direction on the interface between two materials regarding their refractive indices and the incoming light direction. This is done for light entering and leaving a material. Therefore we need the care for sine values bigger than 1.0f, as these will result in numerical errors if not handled. The reason for this occurrence is the total internal reflection.

Advanced Projectiles: Light trail As light traverses pretty fast through the world, we cannot just use a simple capsule for a photon as the players



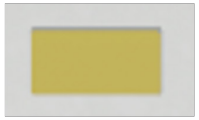
would never be able to see them. Because of the speed of the projectiles, we use a particle system component for gameplay and visibility reasons. This is done by the Unreal Engine's Ribbon Type Data.
This emits some particles following the projectile and connecting them with each other, rendering the trail with some triangles. The result is a visible trail following the light projectile so that players can predict the direction and possibly evade the damage.



Menu: Main, Resolution, Winning, Pause, Join/Start The basic menu has been extended to ensure a complete gameflow. The main menu offers now a resolution option for 640x480, 1280x720 and 1920x1080. Furthermore, a controller help view is added to explain the controller handlings.
In the arena level, the players have to join the game to ensure the correct number of active players are added. Once, all players joined, the first player controller can start the game. During the gameplay, a pause menu can be called, which offers the option to quit the game and return to the main menu. While the pause menu is open, the game itself is set to pause mode.
After all players are killed except of one, a winning screen appears with the option to restart the game or return to the main menu. If the game is restarted, the players need to join again. This makes sure, new players can join as well without the need to go back to the main menu.

Random Map Generator: Use multiple obstacle types In conjunction with the multiple obstacle types the Random Map generator has been extended. It now randomly chooses the type of the obstacle with equal probability distribution.


Glass


Gold


Silicon


Germanium


Diamond

High Target

Visual improvements: Menu, HUD With the intention to make our HUD more appealing and fit better to our game setting, we need to add some visual improvements to make it look more futuristic. Therefore, we decided to use a science-fiction looking font. Furthermore, we added some icons to indicate the health and overheat bars.
Additionally, all menus have been redesigned. The titles of the menus now use a rainbow-colored gradient, representing the color spectrum of the projectiles. The design of the buttons also follows the futuristic setting.
The overall interface is now much more appealing and also follows a clearer design.

Player distance zoom The prior static virtual camera was replaced with a dynamic one. This dynamic camera now adjusts its position according to the position of the players. Also it zooms in, if the players move closer to each other and zooms out, if they separate, respectively.
Since it is important to have a minimum knowledge about the surrounding arena, the maximum zoom-in level is specified.

Smooth Player rotation Currently if the player decides to change the direction of his or her character, the rotation happens instantaneously. To smoothen the movement, the rotation of the character has to interpolate between its current angle and the goal angle.

Detailed Reflections If we want to support more realistic reflections and refractions, we need to consider our energy level of the light projectile and its depending wave length. With this in mind, we can have more detailed effects, so we have added for the different materials additional fix points for wave lengths between 400 µm (blue) and 700 µm (red), in number 7 points, so values in between are interpolated linearly to get the respective refractive indices and extinction coefficients for absorption and damage application. Our information is taken from the side http://refractiveindex.info .

Design revisions

While most game elements currently only provide alpha-status functionality, we did not change any aspect of our initial design decisions.

Challenges

Again as before, we have been faced with some counter-intuitive behaviours and restrictions of the Unreal Engine.

Playtesting

Playtesting Session - Changes - Suggestions

Playtesting session

Playtesting party We planned a Playtesting Party by inviting several people to a Facebook event. 10 guests attended, 3 of them were not familiar with playing video games.
We provided pizza and drinks for the guests and the developers at Force Dev.
At first we welcomed the guests and thanked them for testing our game Lumen Force. We reminded them that we are not testing their skills and only the game. But the guests were eager to start playing, so we let them.
The first several games we observed the players and the game interactions while playing a Free for All mode of 4 players.
After a while, a tournament mode was played to test the competitive aspect of the One on One game mode and its balance.
After a winner was found, we started asking questions while providing the meals.
Finally, as the guests begged us developers to join a tournament, we played the final games with at first Free for All of 4 players and the last few games One on One with 2 players.
The developer team Force Dev claimed the victory.

The party was organized with a Facebook event.

Questions We chose the Play Matrix to find out what the play testers were thinking about the game. As we did not want to include the other games at first, we thought about two matrices and asking for their change of mind.
The play testers thought about the game to be more mentally challenging than needing physical dexterity at first. When we showed the other video games like Starcraft, Unreal and Halo, their decisions switched towards a more dexterity based game.
We noticed there, that we gamers probably like to evaluate our games as being more mind oriented than being a simple dexterity game. However, the play testers might have also been influenced by the shooting aspect of our game and the comparisons.
In general, we think that Lumen Force is a competitive game, needing physical dexterity but also the ability to predict enemy movement and reflection/ refraction directions from the obstacles in the game, so a balance is needed. Therefore we predict our game to be around 80% of Unreal's dexterity factor and Skill (slightly right above Unreal, close to the mark on the right picture).

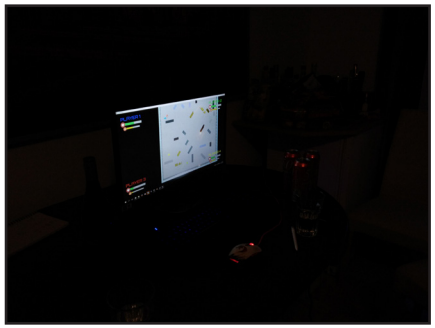
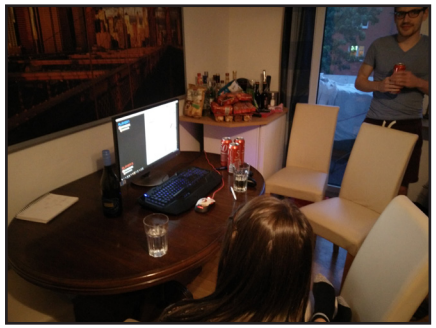
Additionally we asked our guests the Post-Game and Formal Elements Questions on the Playtesting slide. The answers are summarized following the question.

Post-Game Questions

- How did the controls feel? Did they make sense?
  - Intuitive controls, easy to understand and learn
- Did anything feel clunky or awkward?
  - Being stuck shortly on the corners of obstacles
- What was your first impression?
  - Funny arcade game, but missing sounds.
- How did that impression change as you played?
  - Fast to learn, strategy based, skill does help
- Was there anything you found frustrating?
  - Stuck on corners and shooting oneself
- Did the game drag at any point?
  - Not really, One on One slightly slower due to strategy
- What was missing from the game?
  - Sound, more PowerUps
- If you could change just one thing, what would it be?
  - Map size, different (positions of) PowerUps, new modes
- If you were to give this game as a gift, who would you give it to?
  - Friends, family

Formal Elements

- Describe the objective of the game.
  - Survive, Last Man Standing, destroy others
- Was the objective clear at all times?
  - Yes
- What types of choices did you make during the game?
  - Using PowerUps, evade, use cover behind obstacles, do not overheat
- What was the most important decision you made?
  - Attack or hide?
- What was your strategy for winning?
  - Wait for others to kill themselves, take PowerUps, attack overheated enemies
- Did you find any loopholes in the system?
  - No / Push enemies into corner of obstacles
- How would you describe the conflict?
  - Intense
- What elements could be improved?
  - Sounds, random PowerUp positions and duration, less overheat

Impressions from the playtesting party.

Summary

We got valuable feedback and ideas from the play testers.
The game Lumen Force is understood to be a competitive strategy based tactical shooter.
Sustained fire was a preferred tactic in the first few rounds, reflections, refractions, and different energy levels and their colors were not understood.
Players wanted to have a supported tournament mode.
At the 8th round of the game, reflections and refractions on the obstacles were understood and the game was seen as a tactical shooter at the 10th round.
Total internal reflections is seen as a feature and called by the play testers as "mine/bomb has been planted".
Cooldown mechanism and damage increase is used by the more frequently winning person, leading to the understanding of this strategy for the other play testers.

Changes

- Vibration
- Cooldown
- Tweaks & balancing
- Overheat bars' color changes with increasing weapon heat

Planned changes

- Varying map sizes
- Additional tournament mode
- Sounds (background 8/16-bit music, hit sound effects)
- PowerUp to decrease player size (able to walk through close obstacles, harder to hit)

Suggestions

- Different game modes




More impressions from the playtesting party.

Conclusion

Summary - Experience during Class - Conclusion of Class - Video

Summary

Our final game is a strategic, fast-paced top-down shooter, which incorporates the physics of
light into the shot projectiles and destructible obstacles. The art style is kept fairly simple by visualizing only the important elements of the game, so the player keeps the overview, and the theme of the class – 'Arcade' – is fulfilled.
After the playtesting weeks, we fixed several bugs, included a counter before match start, added a high score board, changed the overheat function decrease to linear rather than step-wise, and gave support for fullscreen view.

Experience during class

Initial ideas -> game Our initial game core idea has not changed during the development process. We wanted to create a fast-paced top-down shooter with strategic elements such as the reflection of light as projectiles and the destruction of objects, which is exactly what we did. However, some minor adjustments were made, especially in the destruction concepts. Rather than just splitting the obstacles in two pieces, if a separating line of voxels is destroyed, we now destroy the voxels immediately. Unfortunately, due to the matter of time, we did not add any movement of the obstacles by destructions, but they still could be included in the current concept.
All in all, the major concepts of our game idea worked out pretty well, and just minor modifications had to be done, many of them driven by balancing issues.

Development schedule While we created our development schedule, we have set up a relaxed hour-scheme to prevent trying to squeeze in too many assignments in one week. This also has helped us to figure out the core mechanisms of our game and scheduling them in the first weeks of development.

During the project, it turned out that this strategy was quite effective. For some assignments we needed fewer hours than planned, e.g. the basic destructions, while other assignments turned out to be trickier, e.g. the local multiplayer network environment.

However, we did make several adjustments in the assignments itself, so we merged some assignments together, while splitting up other assignments in more sub-tasks. We also moved some of the tasks in different layers, since we figured out that we misinterpret their importance.

Level of project structure: Contribution or Hindrance? The project structure contributed to our work flow a lot, since we had fixed deadlines for delivering milestones. This kept us working on the game continuously, so that we had a constant progress in our game.

One of the most important aspects, which we consider plays a major role in how our game actually turned out to be, are the first two or three weeks of the project. We put a lot of effort into the brainstorming session, discussed almost up to 10 hours about possible game ideas. Even our final idea came up earlier, we needed this time of discussion and rethinking to realize that this was actually it, what our game should be. The following prototype assignment convinced us even more of our idea, so we started the programming phase with a promising feeling. Additionally, many questions, e.g. defining the core elements, were already set, which supported the programming work flow a lot.

Especially for the organizational aspect, the development schedule was a great support, because everyone could immediately see on which parts the other team members were working on. This enabled everyone to pick an appropriate next assignment in case the other work was already finished, or to address the right person in case of any issues, e.g. bugs or code structure questions.

Conclusion of class

The overall impression of the class is quite positive within our team. The theme 'Arcade' turned out to be more specific than we thought at first sight, but at the end those boundaries were also helpful to find an appropriate idea. Also, since everyone had the same major expectations of an arcade game, we did not have a lot of misunderstandings during the brainstorming process. Additionally, the theme supported us to develop a 'small' game, which suits for the time schedule of the class.

The biggest technical difficulties arose due to the fact that the workflow of C++-files within the Unreal Engine is not so well documented as the workflow with blueprints. Also, since some of us worked with the Engine for their first time, it took quite a bit to get familiar with the Engine specific classes and working environment. If we would now program a next game, we probably could realize some aspects in a more elegant way.

Since our game focused since the beginning on an exciting gameplay mechanism while keeping the amount of level design and art design aspects quite low, we could mainly work in our expert fields. This further led to the fact that we could meet all of our main milestones. Only the additional feature aspects, such as different game modes, e.g. a cooperative mode, is not realized, since it would include a whole new bunch of tasks which were simply not manageable during the given time frame. However, those additional features were clearly distinguishable from the core ideas, so it was pretty easy to cut them off from our project plan.

On the other hand, there are some improvement suggestions for the class we have. First of all, sometimes the expected deliveries for the milestones were not quite clear. One milestone e.g. required only optional slides, but in the class slides were actually expected due to the fact that out-of-class visitors were participating – this information was not given beforehand.

After that the delivery requirements were changed, stating that a presentation should be given, nothing mentioned about the slides. However, since then slides were actually mandatory, which was not stated clearly.

As a conclusion, the class was in our point of view a success, providing us at the end even with a great self-made game. However, some milestones deliveries could be stated more clearly.

Video

To highlight the major aspects of our game, we created a trailer-like video clip, which can be found here: https://youtu.be/W9yLjXaKimE