Team 2 – Escher's 11

Game Programming Laboratory

# THAT FAILED BANK ROBBERY

*„Should Have Planned It Better "*

| | |
|---|---|
| Simone Guggiari | – PRODUCER + GAME DESIGNER + GAMEPLAY AND ENGINE PROGRAMMER |
| Nicolas Huart | – GAME DESIGNER + LEVEL DESIGNER + MENU PROGRAMMER |
| Alexander Lexus | – GAME DESIGNER + PHYSICS PROGRAMMER + PROCEDURAL CONTENT |
| Andreas Emch | – GAME DESIGNER + PHYSICS PROGRAMMER + EFFECTS PROGRAMMER |
| Xingze Tian | – GAME DESIGNER + AUDIO ENGINEER + GRAPHICS PROGRAMMER |

# Chapter 1. Formal Project Proposal

## 1.1. Game Description

### 1.1.1. Logline

Two teams of clumsy robbers inside unlikely vehicles must collect money and valuables scattered all over Credit Suisse and disrupt their opponents before the police arrives after a badly timed explosion sabotaged their master plan.

### 1.1.2. Overview

'That Failed Bank Robbery' is a competitive local multiplayer game for 2 or 4 players in which two teams of thieves try to rob the same bank at the same time. The goal of the game is to collect the most money before the round ends or the getaway vehicle is destroyed. Each player in the game controls a vehicle and can collect money by opening vaults, cracking crates or collecting valuables present on the map. Players can also steal their opponent's loot by attacking them or their base. The game is set inside Credit Suisse and features a wide array of power-ups and different valuables and riches. The more money a player collects before bringing it to its base, the riskier the play becomes, as vehicles move slower and have impaired attack when loaded. This provides a fun layer of quick and dirty action on top of a more strategic game in which several tactics can bring to victory.

The game is developed by Simone Guggiari, Nicolas Huart, Alexander Lexus, Andreas Emch and Xingze Tian as a project for the Game Programming Laboratory offered at ETH Zürich in the Spring Semester of 2018 under the supervision of Prof. Robert Sumner.

### 1.1.3. Guiding Principles

We have three principles we want to base our game around. These are that our game should be:

- FUN
- SIMPLE
- BEAUTIFUL

We believe that to have a compelling game that can be enjoyed by players as soon as they jump in, we need to provide a simple game with an intuitive control scheme as well as clean interface. We also need a strong fun component, that provides a layer of strategy underneath the frenetic action-packed gameplay. The game rules should be easy to learn yet provide emergent gameplay to keep the game fresh. All of this should be wrapped in a game and a user interface that is both beautiful and attractive, as well as polished.

Therefore, our game should be nice to look at, easy enough to let players jump right in and fun enough to keep them coming back, and have some layers of depth and strategy to keep the player engaged even after playing a few rounds.

### 1.1.4. Background Story

The story begins in Escher's time, in 1856 as he was founding Credit Suisse. We see him building the bank from scratch with nothing but hard, honest work. Fast forward 163 years, the year is 2019 and Credit Suisse is now a giant in Swiss economy and attracts all kind of people and business. Two teams of robbers, after learning of Escher's history, decide to pay him homage by robbing his bank on the 200th anniversary of his birth. Escher's got rich with honest work, and now it's time to get rich with honest *dishonest* work! They plan a grand entry in the main building to blow up the vault, which is fully packed on this special day. However, due to bad planning, the explosives go off too early and money is scattered all over the bank. The police are on their way and the robbers have limited time to gather all the valuables and get out of there before it's too late. They jump into the first vehicle they can find, a forklift used to move stuff around, and duel to death to be the ones that get away with the most cash.

### 1.1.5. Design Decisions

#### 1.1.5.1. Uniqueness

Our game strives to be unique by providing an innovative type of gameplay which is not found in most titles. We decided to stay away from platformers and shooters, and instead combine elements of action, racing and strategy games together, as well as some elements typical of party games. The final experience we try to accomplish is a fast-paced action racing game with combat elements, as well as different strategies and tactics that allow to reach the goal of the game.

#### 1.1.5.2. mechanics

The main game mechanic allows players to control their vehicles around the map, to bump into other players and obstacles like a bumper car and destroy crates and stunning the opponents by performing a dash. By going over a valuable, it is collected and added to the inventory, and in a similar manner powerups can be picked up and used.

#### 1.1.5.3. Settings

The game is set inside of Credit Suisse, in the aftermath of an explosion. Money is scattered everywhere, and more valuable items are found on the upper part of the map, near to the vault, jewelry and safe boxes. The map is rectangular, with the two bases (getaway vehicle) on the lower side.

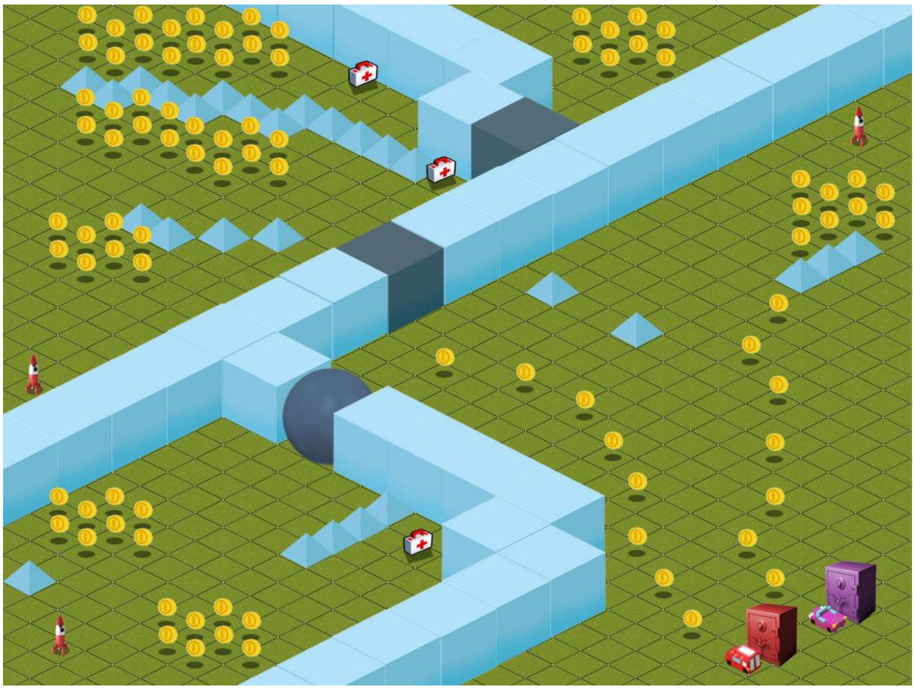
#### 1.1.5.4. Look and feel

We plan to have a cartoonish, colorful and polished look to our game. The models will be low-poly, and this will allow us to have more detail in our scene. The models will fit the theme.

The objects and model are 3D, however most of the action will happen on a flat plane for simplicity and realism.

The screen is split into the number of players, each part showing a player's view (one camera for each player). There is also a small global map, where each player can see where the coins, the obstacles and the power ups are.

The camera looks at the map from an angle, smoothly follows the player and provides an isometric look. Here is a mockup of our view.



### 1.1.6. Audience, Platform and Marketing

The main audience for the game are casual player, such as those that will play the game at our booth at the end of the semester. The game is also geared towards players with more experience by providing additional challenges.

The game will be developed on windows and deployed on the Xbox One. It will be possible to play with the controllers on both windows and Xbox.

We plan to publish the game for both platforms at the end of the semester as our extra target.

We will be marketing the game mainly via a website that one of our members will setup, as well as other channels once the Lab is finished.

### 1.1.7. Game Elements

Here we describe some of the main elements that will be found in our game

#### 1.1.7.1. Winning Conditions

The goal of each player is to maximize his profit by bringing money back to his base. The game ends when the time limit is reached or when one of the player manages to destroy his opponent's base. Players can combine different winning strategies for a common goal: collecting the most coins.

#### 1.1.7.2. Characters

Different vehicles will be available in our game. The first one will be a forklift, which is well suited to move around objects and attack other vehicles with his claws. We will also implement two other vehicles for a total of 3, with different stats such as capacity, speed and attack. They will either be two other variations of forklifts, or more 'exotic' types of vehicles such as street sweepers, excavators or something similar as they can both carry material as well as attack.

Each player in the game controls one of these vehicles with specific life, speed and capacity. Each player starts at his own base.

#### 1.1.7.3. Dash Attacks

Players can attack each other by performing a dash. When one player is hit by the other, with a certain probability he will become stunned and lose some of his money and life. During this time the other player can either collect the money and run away, or stick around for more action. When players have their whole life depleted, they lose all the coins they were carrying and respawn in their base after a small delay. Dash attacks can also be used to open crates but cannot damage bases or vaults.

#### 1.1.7.4. Valuables

Our game will feature different types of valuables that can be collected and brought back to the base. Each will have a weight and a value (like the knapsack problem). Items found in the beginning of the map will have less value (such as bills or coins), while some found later (jewelry, gold) will have a higher ratio of value per weight, making them more interesting to bring back to base albeit riskier. Whenever a player goes through a place with a valuable, he collects if it is possible (still has inventory space). A player cannot collect more coins than the capacity of his vehicle allows. If he manages to go back to his base, his total score gets increased by the total value he just collected. If on the way, he loses all his life, he has to restart from his base and all his collected coins are lost, which will be scattered around the place where he died. The coins are initially randomly placed over the map and new coins will appear each time a player brings a coin back to his base or after some time, according to our procedural generation algorithm.

#### 1.1.7.5. Power ups

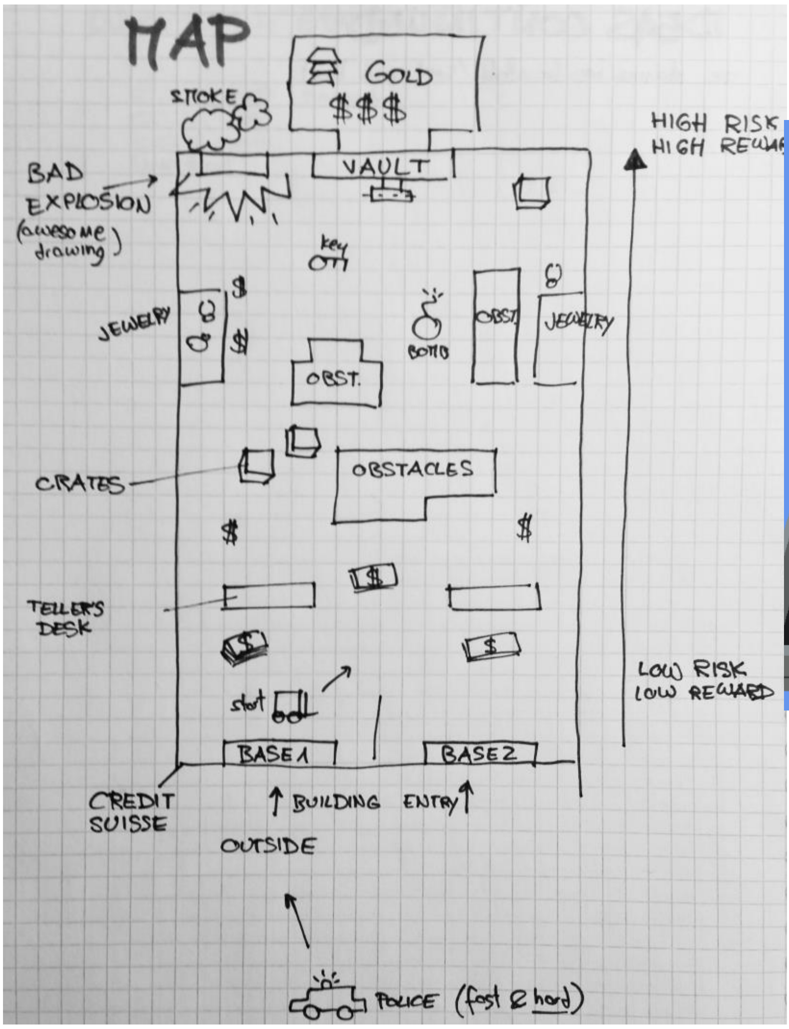
We plan to implement a few power-ups in our game. Power-ups change the property of each character and can be either temporary (e.g. speed boost) or permanent (e.g. capacity boost, medic kit). There is also a special power-up (bombs) that can be used to damage obstacles, open vaults or damage the enemy player's base. Keys will also allow players to get inside the vault. Other power ups might be used to improve the defense level of the base.

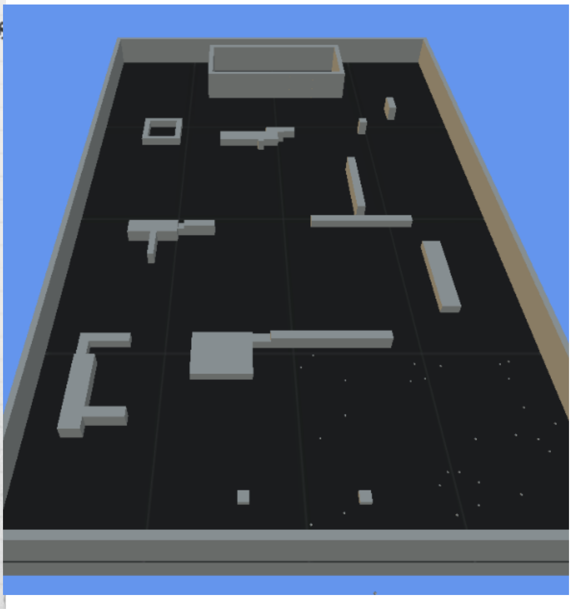
#### 1.1.7.6. Crates and Vaults

Scattered in the map, we will have special crates that can be cracked by performing a dash towards them. Once cracked, the crates will reveal valuables and possible powerups that they had inside.
Vaults on the other hand will be found at the top of the map, in the riskiest place, and it will be possible to open them either with a key or a bomb. After this the player will be able to break in. Once opened, they will reveal a lot of valuables to be collected. This will be a very risky action that a player can choose to undertake, but the potential payoff will also be great as vaults will be filled with money and gold.

### 1.1.8. Map

Our map will be rectangular, with the entry of the bank on the bottom together with the team bases, and will contain everything that could be found inside a bank, from teller's desks to jewelry, safety boxes, obstacles and vaults. The drawing attached should give an idea of the layout. The further up the player decides to go, the riskier the play becomes.





## 1.2. Big Idea Bullseye

- *Collect procedural coins on the map or fight your opponent*
- *Strategic layer on top of fast-paced fun action*

BIG IDEA BULLSEYE
COLLECT PROCEDURAL LOOT AND FIGHT
STRATEGIC FAST-PACED ACTION

## 1.3. Technical Achievements

### 1.3.1. Rigidbody Simulation

All the players will control a vehicle, which will be modeled as a rigidbody simulation. This includes a model for forces, torques, and velocities, making sure that no vehicle can go inside objects (collision detection and resolution). We will also implement friction and restitution coefficients to have cars bounce away when colliding with something, such as bumper cars do. The dash attack will also be modeled as a rigidbody simulation (imagine air hockey), and will strive to keep an arcade feel to our game controls overall by tweaking all the physics parameters.

### 1.3.2. Procedural Generation

We plan on implementing a procedural generation algorithm that will take care of placing coins, power-ups and obstacles. Doing so the map will always be different and thus it should be more fun and variated to play. Things to be procedurally generated include coins, powerups, crates and obstacles. Our procedural generation algorithm will be smart enough to try to balance the game, meaning that will spawn more valuables where the risk associated with them is proportional to the reward, will try to spawn and favor the player which is currently losing, and will decide when is the best time to spawn one of the most powerful powerups. We will also procedurally generate obstacles inside our map making sure every area is still accessible.

## 1.4. Team

In this section, we present the responsibilities that each member of our team will take upon himself.

### 1.4.1. Simone

Simone will take care of the game engine, as well as the organization of our software structure. He will mainly be involved in programming the gameplay features as well as simple modeling and graphic tasks. He will also make sure that the team is following the project schedule.

### 1.4.2. Nicolas

Nicolas will be in charge of the design of the static part of the map as level designer. He will work on the menu design and implementation. He will also be involved in the character modelling and will be managing the library of assets.

### 1.4.3. Alexander

Alexander is building with Andy the rigidbody simulation to handle the physics correctly and testing the game to make sure it’s fun to play. He is also in charge of the dynamic part of the map as placing special power ups, coins and other dynamic obstacles procedurally. In addition, he is helping out with the visual appearance of the level and the total project. In terms of side tasks, he is working on the slides and the Html pages for marketing the game.

### 1.4.4. Andreas

Andreas will implement the rigid body simulation as well as the collision detection part together with Alexander. This includes physical simulation, collision-handling, spinning wheels and friction, as well to integrate it into the game-play. Additionally, Andreas will be controlling visual effects such as shaders, lightning, particle effects, etc.

### 1.4.5. Xingze

Xingze will be in charge of the sound effects (background music, sounds triggered by actions, start and end of game sound effects) and sound library. She will also take care of all the models, textures, light maps and other needed assets. In addition, she will implement lighting effects. In the final steps, she will be creating the trailer of the game and preparing slides with Alexander for the presentation.

## 1.5. Development Schedule

We will be following an agile schedule, that consists in small sprints of one week in which each team member has one well defined task to complete (or multiple smaller ones). We will have weekly meeting to discuss the current achievements and decide the tasks for the following sprint, as well as test the game, discuss new ideas and make sure we are on schedule.

### 1.5.1. Layered Task Breakdown

Our high-level view for the layered task breakdown is as follows: in the functional minimum, we plan to work mostly on tools for the engine and gameplay. This should give us a basic playable game. In the following phase, we extend gameplay functionality and add most of the features we want to have in our finished game. At the end of this stage we plan to have a fully working game that although is very rough, allows us to play. In the next phase (desired goal) we plan to focus mostly on graphics and menu, making the game something pleasant to look at, as well as including all the graphic assets and required menus. We include all the polishing in the last phase, such as audio, game balancing, and effects.

#### 1.5.1.1. Functional Minimum

Our goal for the functional minimum is to have a basic game in which the player can control his avatar, move around in the level while picking up money and bring it to his base, with an isometric camera smoothly following the player. We want to have a simple HUD showing statistics such as time remaining in the round and money collected so far. No physical simulation will be present yet, and the level will just be a simple plane. The goal of this phase is to get everybody accustomed to working in MonoGame and have something we can start experimenting with. We also plan to start experimenting with technical stuff such as physics and rendering, producing a simple 2D rigidbody controller. We plan to be able to deploy this build to the Xbox already to make sure we don’t run into technical issues later.

- Game engine:
  - Implemented game objects with 3d transforms and components
  - Implemented classes for camera, audio, input, scene, prefabs
  - Implemented classes for utility, time, coroutines, basic physics
  - Drawing 3d models as well as pipeline loading
  - Game running on Xbox
- Gameplay
  - Simple controller to move vehicles around
  - Camera following player smoothly
  - Pickup money, bring to base
  - Round time, winning condition
  - Simple HUD
  - Basic primitive level

#### 1.5.1.2. Low Target

Our low target includes extending the game to allowing a second player to compete. This includes the addition of split screen functionality, ability to perform attacks and cause damage and money loss, respawn. We plan to implement a basic primitive level in which collisions work and to improve our player control to work with rigidbodies. We plan to expand the gameplay with almost all of the features, as well as expanding the HUD and start balancing the game to increase the fun factor.

- Game engine:
    - Physics fully implemented
    - Control implemented with rigidbodies
    - Procedural spawning
- Gameplay
    - 2 and 4 player split screen
    - Attacks
    - Damage/death/respawn
    - Stun/money loss
    - Level with primitives and collision

#### 1.5.1.3. DESIRED TARGET

In this phase we start focusing on graphics and menus. We plan to have a working game already, now it’s time to have a level with all graphics assets, menu that allows to select a starting avatar, add advanced gameplay functionalities such as power-ups, crates and vaults. We start distinguishing players by vehicle type with different stats such as speed and capacity. We will have some basic procedural money generation that makes the gameplay more unpredictable and thus fun. We will start having a small library of sounds we want to add as well as implementing most of them in the game.

- Gameplay
    - Vaults
    - Crates
    - Base damage
    - Power-ups
- Interface
    - Embellish and power-ups usage
    - Mini-map
- Menu
    - Menu windows implemented (main/play/join/options/…)
    - Menu transitions and effects
- Graphics
    - Added graphic assets
    - 1st level finished
    - 3 vehicles modeled
    - 2d art/title/tutorial
- Audio
    - Basic audio and music

#### 1.5.1.4. HIGH TARGET

For the high target we will mainly focusing in improving the existing game. The game’s graphics will be enhanced with additional work on the visual effects, such as shaders and particle effects (e.g. for dust). To improve the replayability of the game, other levels will be added.

- UI and Menu
    - Additional polish
- Audio
    - All sounds and music

- Effects
    - Particles (dust/sparkles/explosion)
    - Post-processing
- Game
    - 2nd level finished
    - Balance and polish
    - Final trailer and presentation

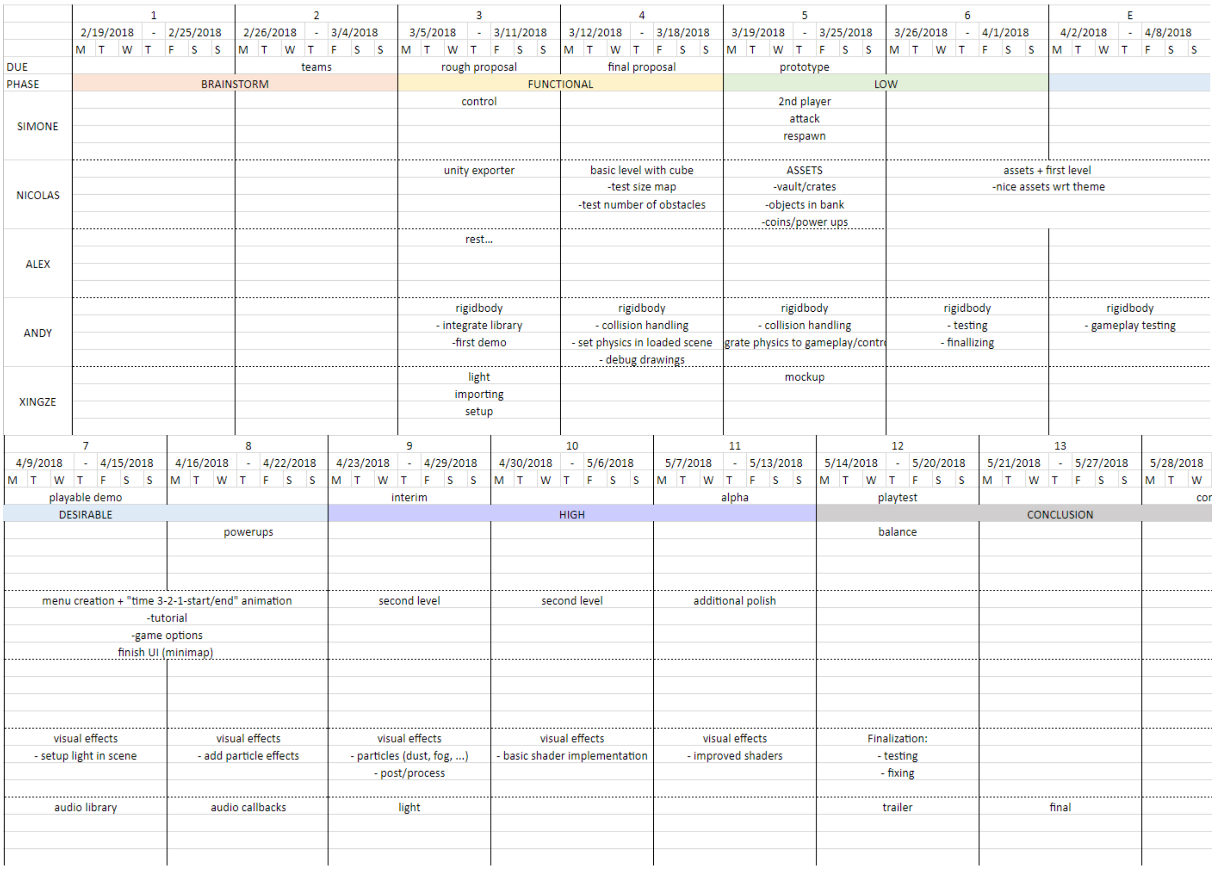
#### 1.5.1.5. Extras

Things we would like to have in our game but know that will not be able to implement in the limited timespan of one semester are a single player mode, in which the computer controls one team of robbers. This includes AI, navigation, decision making and strategic planning. Implementing a more advanced progression mode that is persistent between rounds of the game would also be nice. We would also love to be able to publish our game on different stores, start marketing it with a website and have one article written about our game.

- Single player
    - Enemy AI (decision and strategic planning)
    - Navigation
- Gameplay
    - Other powerups
    - Buyable from store between rounds
    - Persistent between levels
- Publish
    - Store
    - Website / article

### 1.5.2. Timeline

This is a timeline showing the whole semester divided into the 4 phases we described. It is a high-level overview of which tasks will need to be done when.



| | 1 | 2 | 3 | 4 | 5 | 6 | E |
|---|---|---|---|---|---|---|---|
| | 2/19/2018 - 2/25/2018 | 2/26/2018 - 3/4/2018 | 3/5/2018 - 3/11/2018 | 3/12/2018 - 3/18/2018 | 3/19/2018 - 3/25/2018 | 3/26/2018 - 4/1/2018 | 4/2/2018 - 4/8/2018 |
| | M T W T F S S | M T W T F S S | M T W T F S S | M T W T F S S | M T W T F S S | M T W T F S S | M T W T F S S |
| DUE | | teams | rough proposal | final proposal | prototype | | |
| PHASE | BRAINSTORM | | FUNCTIONAL | | LOW | | |
| SIMONE | | | control | | 2nd player<br>attack<br>respawn | | |
| NICOLAS | | | unity exporter | basic level with cube<br>-test size map<br>-test number of obstacles | ASSETS<br>-vault/crates<br>-objects in bank<br>-coins/power ups | assets + first level<br>-nice assets wrt theme | |
| ALEX | | | rest... | | | | |
| ANDY | | | rigidbody<br>- integrate library<br>-first demo | rigidbody<br>- collision handling<br>- set physics in loaded scene<br>- debug drawings | rigidbody<br>- collision handling<br>grate physics to gameplay/contr | rigidbody<br>- testing<br>- finallizing | rigidbody<br>- gameplay testing |
| XINGZE | | | light<br>importing<br>setup | | mockup | | |

| 7 | 8 | 9 | 10 | 11 | 12 | 13 | |
|---|---|---|---|---|---|---|---|
| 4/9/2018 - 4/15/2018 | 4/16/2018 - 4/22/2018 | 4/23/2018 - 4/29/2018 | 4/30/2018 - 5/6/2018 | 5/7/2018 - 5/13/2018 | 5/14/2018 - 5/20/2018 | 5/21/2018 - 5/27/2018 | 5/28/2018 |
| M T W T F S S | M T W T F S S | M T W T F S S | M T W T F S S | M T W T F S S | M T W T F S S | M T W T F S S | M T W |
| playable demo | | interim | | alpha | playtest | | cor |
| DESIRABLE | | HIGH | | | CONCLUSION | | |
| | powerups | | | | balance | | |
| menu creation + "time 3-2-1-start/end" animation<br>-tutorial<br>-game options<br>finish UI (minimap) | | second level | second level | additional polish | | | |
| | | | | | | | |
| visual effects<br>- setup light in scene | visual effects<br>- add particle effects | visual effects<br>- particles (dust, fog, ...)<br>- post/process | visual effects<br>- basic shader implementation | visual effects<br>- improved shaders | Finalization:<br>- testing<br>- fixing | | |
| audio library | audio callbacks | light | | | trailer | final | |

### 1.5.3. Task List

This is a more in depth and accurate list in which tasks are subdivided by category (gameplay, engine, menu, interface, graphics, …) and by phase, as well as who will be making them. Each category has one or two responsible. As this excel table is quite big, please look at it on the next page. We didn't write an expected number of hours per task, but balanced them in such a way that we will need to follow the weeks marked above. In our estimate, each member of the group will be working around 25 hours weekly to accomplish all of the marked tasks.

## 1.6. Assessment

We will consider our game a success if we manage to get a fun, simple and polished experience out of it. We believe that the most interesting part of our game is the possibility of competition that will spark challenges to arise, as well as the possibility to coordinate and communicate to reach the common goal. If we also manage to have some emergent gameplay dynamic arise from our simple set of rules, it will be another victory for us, as well as having a beautiful to look at videogame that people have fun playing and that will make them keep the controller for "just another round".

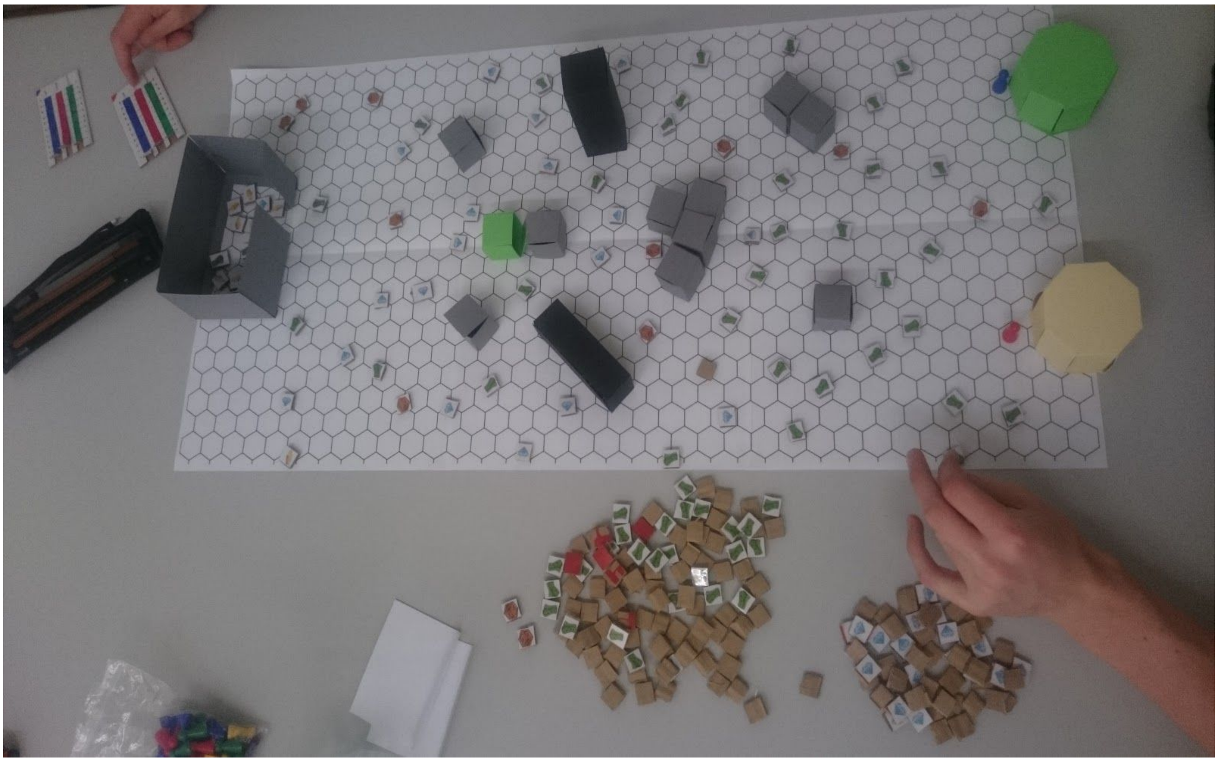
|  |  |  | FUNCTIONAL | LOW | DESIRABLE | HIGH | EXTRA |
|---|---|---|---|---|---|---|---|
|  |  |  | An ugly functioning 1player prototype in which a vehicle can be controlled around the level, pick up money and bring them to the base, showing a timer for time left and money accumulated | Add 2-player functionality, including attacks/lose coins and respawn, and more elaborate camera control. Implement collisions/rigidbodies and unity level exporter | Add powerups as well as base damage and vault. Put graphics and sound in place, start working on a second level and nice UI/MENU. Improve feel and balance | Polish game, add menu selection, balance, make fun, | TODO: add stuff |
| WEEKS |  |  | W3 - W4 | W5-W6 | WE-W8 | W9-W11 |  |
| GAMEPLAY | control | S | simple joystick to move character<br>2nd player<br>attack | 4 players<br>leave cash | improve feel<br>integrate physics |  |  |
|  | camera | S | one screen following player<br>split screen | camera rotation | camera fly through<br>implement splines |  |  |
|  | interaction | N+S | pickup money<br>bring to base<br>life/death/respawn<br>lose coins | base damage<br>vault<br>crates |  | broken effects? | bots<br>navigation |
|  | abilities | N+S | slower the more money | powerup usage<br>opening vault (timer/explosion) |  | buy powerups<br>different player stats |  |
| TOOLS | engine | S | basic structure<br>gameobjects+components<br>transforms<br>drawing/loading | be able to show video<br>scene loading? |  |  |  |
|  | physics | D+A | unity working of our goal<br>basic oncollisionenter callback | collision detection<br>rigidbody 2d<br>basic primitives for collision |  |  |  |
|  | deploy | D+S | test working on xbox<br>make sure everyone compatible<br>git working<br>windows developer account | continue deploying<br>merge branches |  |  |  |
| ASSETS | models | N | primitives exported<br>look at assets | purchased assets & imported<br>1st vehicle modeled | modeled 3 vehicles<br>modeled other unique assets |  |  |
|  | level | N | simple primitives 3D<br>plane<br>unity export tool | more advanced primitives<br>scene made in unity<br>improve procedural spawning | integrated models<br>1st full level<br>procedural generation implemented | 2nd full level<br>procedural generation improved |  |
|  | images |  | basic images for report | rough drawing of what we need | create tutorial image | implement 2d art |  |
|  | artist |  | decide budget | start contacting artist<br>come up with art that we need | get 2d art |  |  |
| INTERFACE | UI | N+S | timer for round<br>money in base and carried<br>health / attack | basic minimap | Embellish ui<br>powerup UI |  |  |
|  | menu | N | plan look and storyboard | basic gameover/restart | menu windows implemented<br>basic character selection | additional menus added<br>menu transitions |  |
| POLISH | sound | X | get familiar with how it works<br>play in 3d | list of sounds<br>library of sounds<br>selected music | added sound | more sounds |  |
|  | light | X | lightmaps in unity baked<br>familiar with lights in monogame<br>pointlight/spotlight | works in monogame | polish lighting |  |  |
|  | particles | D | basic engine working 2d | working 3d<br>list needed particle effects | create needed particle effects |  |  |
|  | shaders/materials | X+D | how to use materials<br>mockup of final desired graphics |  | have post processing working | implement post process stack | custom shaders |
| DELIVERABLES | docs | - | rough proposal<br>formal proposal | create paper prototype<br>improve it |  |  |  |
|  | slides | X | create slides for presentation<br>create pictures/drawings |  |  |  |  |
|  | video | X | think of story | trailer script<br>trailer storyboard | basic trailer cut | polished trailer |  |
| OTHER | fun | - | brainstorm<br>meetings/logistics | fun factor | balance |  |  |
|  | debug/test | - | read code and get familiar | always | always | do a lot |  |
|  | relations |  | talk with previous groups<br>get code | talk with studio gobo |  |  |  |
|  | marketing | A | think about webpage | setup basic webpage/blog | added graphics/text | polished/published<br>webpage finished | sell game on store<br>get some website to talk about |

# Chapter 2. Prototype

## 2.1. Prototype Setup

### 2.1.1 The Basic Model

We set up our prototype based on the game rules described in the last section with the following objects to represent the key features:



#### 2.1.1.1 Characters

Characters are represented as little car models or small chess piece (grabbed from an existing board game). The small pieces are in different colors for players to choose.

#### 2.1.1.2 The Map

The map of the game is represented as a rectangle paper with about 30cm x 120cm in size (3 A3 paper sticking vertically). The size of the map has been experimented to ensure players can meet each other as well as being able to explore further for more money. In addition, honeycomb grid is used to enable players to move in all directions and to control the speed of movements.

#### 2.1.1.3 The Base

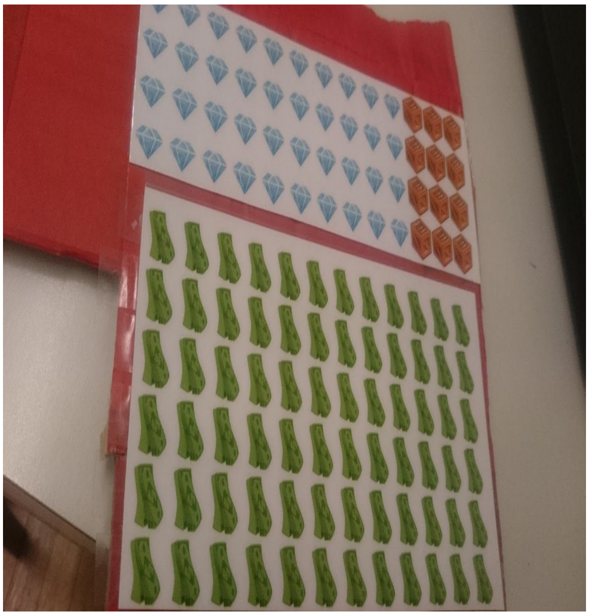
The two bases (the yellow and green polygonal columns) are located at the bottom of the map, and this is where both the players start the game. The two bases are close to each other to ensure fairness, but not too close to avoid making it too easy for each player to attack the other's base.

#### 2.1.1.4 Crates and Vaults

There is a vault located at the top of the map (as shown as the grey open box) which requires a key to open. In addition, crates are represented using small card boards with a box symbol on it. Players can choose to attack these crates to randomly get additional valuables and power-ups.

#### 2.1.1.5 Valuables and Power-ups

We use small paper boards with different stickers to represent the valuables (cash, diamond and gold) and power-ups (capacity, life, speed) which are spread randomly around the map. At the bottom of the map, more cash are placed, while there are more diamonds and gold present near the vault.



#### 2.1.1.6 Randomness

To simulate the randomness process dices are used. Every time players attack each other, a dice will be thrown to determine how much money he/she will lose. Similarly, players might get power-ups or valuables after attacking crates depending the dice throwing result.

#### 2.1.1.7 Player Stats

We use a card for each player to record their capacity, life and stamina. The statistics are represented as progress bars which can be slid on the card.

### 2.1.2 How It Works

The prototype is played by two people, with an additional person acting as the computer. The prototype uses rounds for movement control and efficiency. The prototype is designed to play within 30 rounds. At each round, the players choose its action (moving / attacking) with respect to the following rules:

#### 2.1.2.1 Initialization

At the first round, each player has 100% health, 100% stamina, a capacity of 10 for valuables and a capacity of 3 for power-ups. For this prototype, we consider all the objects take the same amount of capacity (i.e. the maximum number of objects a player can carry during the play is 10).

#### 2.1.2.2 Movement

In each round, every player can move a certain number of tiles in any direction they like and collect all the valuables on the path:

- Remaining capacity >= 5 => 6 tiles
- Remaining capacity >= 2 => 4 tiles
- Otherwise => 2 tiles

#### 2.1.2.3 Attacking

Players can attack each other if they are next to each other. At each round, if a player's stamina is over 50% and his/her opponent is nearby, he/she can make an attack by saying "shhh". After doing so, independent of how much capacity is left for the player, he/she can move 6 tiles in a straight line and his/her stamina decreases 50% immediately. The attacked player will lose 50% of his life and a random amount of money decided randomly by the computer. If a player loses all his life, he is forced to get back to the base and loses all the money. He will start the next round with full life.

Players can also attack loot-boxes or crates when passing them. After being attacked, the box gets destroyed and new collectables are placed around the position where players can collect in the next round.

#### 2.1.2.3 Valuables

At each round, players can collect valuables when they pass a tile with a collectable item and the player still has remaining capacity. Once a player reaches full capacity, he cannot collect anything on the way until he/she returns the base and unloads everything. When players have capacity smaller than 10, they can unload the items on the current position as an offset for speed (if the current position is not the base, they lose the items).

#### 2.1.2.4 Power-ups

When players have remaining power-up capacity, they can pick up the power-up when passing by. Each power-up takes up one space. In the prototype we support 4 types of power-ups:

- **Bomb**: it can be used to destroy the base of the opponent. To use a bomb, simply place it at the desired location, and it will explode in 2 rounds. A base requires 2 bombs to destroy, once destroyed, the player who placed the bomb gets all the items in the base
- **Capacity**: it can be used to expand the capacity by 10. This action is permanent, i.e. if a user has capacity 10, after using this power-up, he/she will have capacity 20.
- **Health**: the health of the player will increase by 50% (cannot exceed the maximum 100%)
- **Boost:** the speed of the player increases to 6 tiles for 2 rounds

#### 2.1.2.5 Computer Control

The person acting as the computer counts the number of rounds, notifies players the start of each round (by knocking the table) and reminds them when the police comes (when only 10 rounds are remaining).

The computer updates players' statistics at each round including increasing stamina by 10% if no attack happened or decreasing by 50% if an attack occurred. He/she also needs to modify capacity accordingly, updating the bomb-counters if placed.

When users attack each other or the loot-boxes, the computer throws the dices to determine how much money a player wins or how many new collectables are generated. The computer also needs to randomly place new loot-boxes, he/she first throws a dice to decide if a new loot-box should be placed or not, then he/she places the loot-box near the less-privileged player.

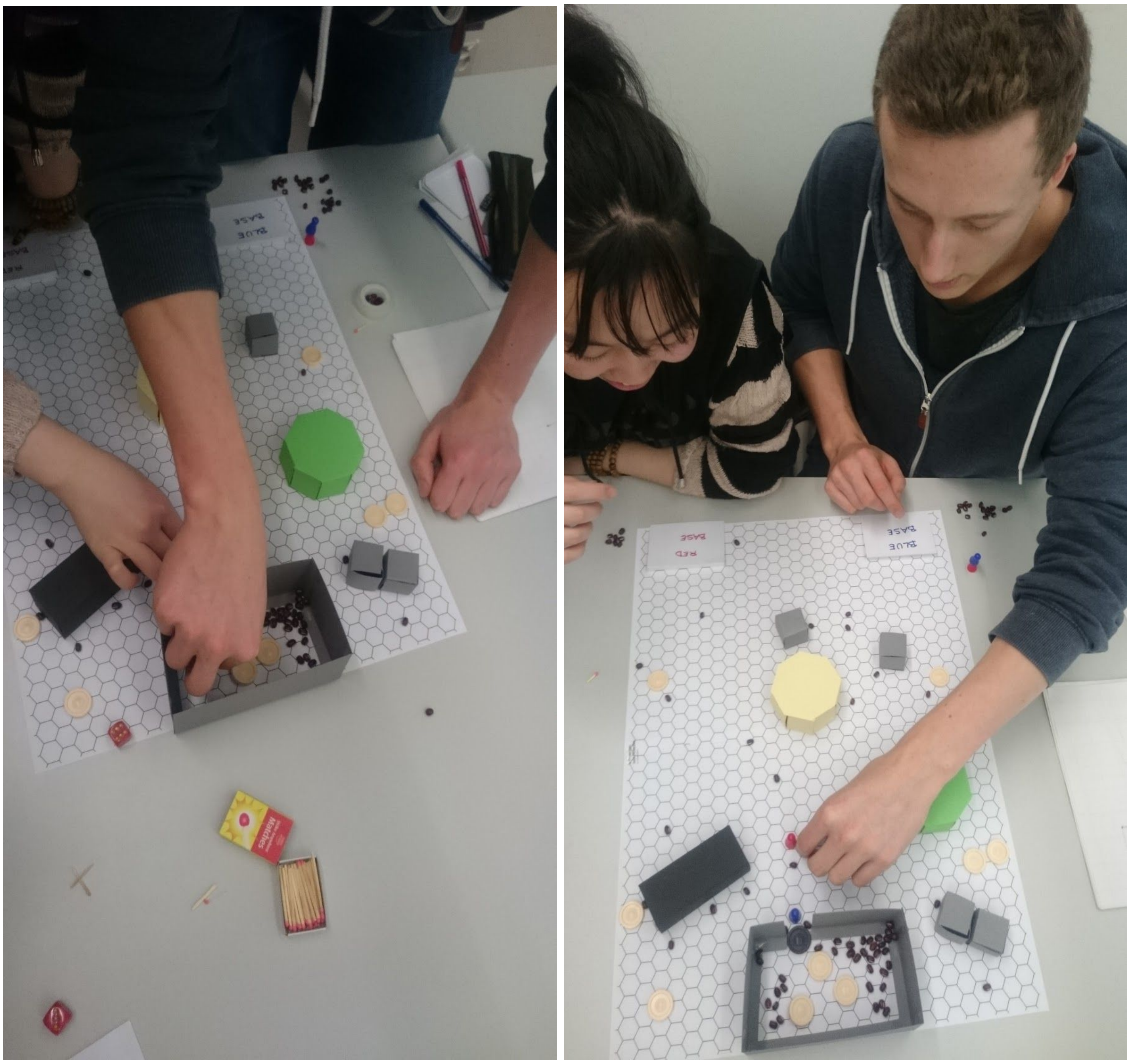
#### 2.1.2.6 End of Game

At the end of the game (at the 30th round) the police come and all the players should return to their bases. The player fails to do so loses the game. If both players succeeded returning to their base safely, the valuables collected at the base will be counted where cash counts as 1 point, diamond 3 points and gold 5 points. The player with more points wins the game.

## 2.2. Playing Experience

The prototype was modified and improved during the construction and playing process.  During the playing, we experimented on the map size, types of objects that can be used to simulate the features, the speed of movement and how power-ups should be used.

(Initially we used beans as coins, but they are rolling around all the time!)



After several experiments and improving on the prototypes, we believe we have finally reached a state that the prototype is fun and attractive to play! The number of rounds is reasonable, the size of the map is appropriate and the amount of randomness also increases the entertainment of the game.

At the start of the game, the person works as the computer places everything on the map randomly, then the game starts. At every round, each player needs to balance his/her eager of money and the risk he/she takes. Sometimes it is also necessary to give up some of his/her treasures to escape from an attack or get a faster speed. In this prototype, how each action is selected and performed is super fun. In addition to playing against each other in the game, the interaction between the players in real life while playing the game also adds further entertainment to the game.

What's more, the randomness has added another taste of mysterious and fun to the game. In a particular game, one of the player has much better luck than the other even the computer has tried his best to balance, this brings the game a bit casino feeling.

## 2.3. Findings and Conclusion

During the process of creating and playing the prototype, we have gained much more experience in game design, here are a few findings we would like to point out particularly:

- **The things we omitted are always more than we expected:** before we started the prototype, we have planned all the rules and objects we needed. During the actual play, we can always find situations for which our rules are not specific enough.
- **Feature parameters require a large amount of experiments and user surveys:** to find the most balanced parameters of the features, for example, the size of the map, the speed, the capacity, etc. The optimal never comes for free and can only be retrieved or getting close to by experimenting, modifying and improving.

In summary, prototyping plays an important role in game design, it helps us to perfect the game mechanism, to find the most suitable parameters, and the most importantly, we gain new ideas through the playing process, which helps us to design and create a much more interesting and enjoyable game.

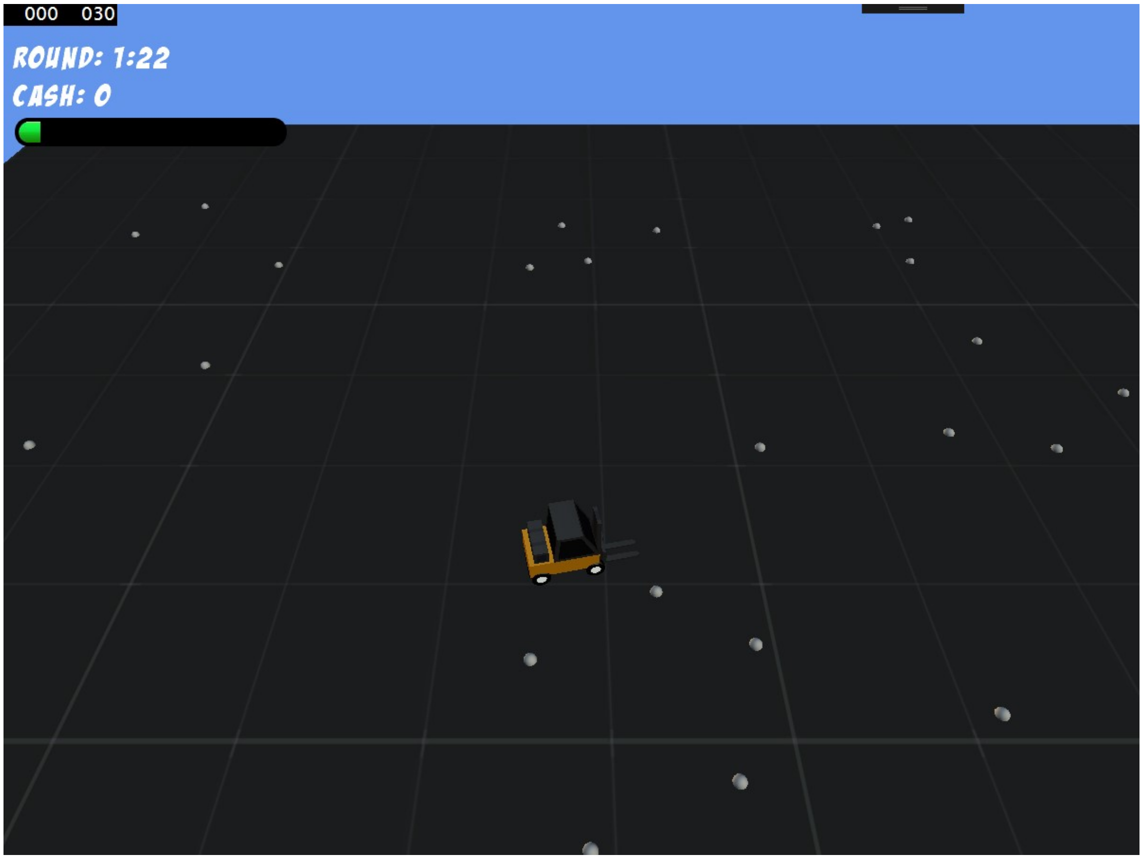
# Chapter 3. Interim Report

## 3.1. Progress

At this point in time, we have finished the functional minimum, low and desirable targets and are now working full time into the high layer and even started some of the features in the extra target.

### 3.1.1 Functional Minimum

We achieved functional minimum targets in the 4th week of the project. We created a functioning one player prototype in which a vehicle can be controlled and moved around. It is also possible to pick up cash and bring it to the base, increasing the total balance. At this point we also had the vehicle velocity depending on the capacity and rough cube graphics. Here is a screenshot:



#### 3.1.1.1 Gameplay

At this stage, users can use the joystick or keyboard to move the vehicle and pick up money scattered on the plane. The more money the user collects, the slower it becomes. Players can return to the base and unload the money, thus increasing the balance.

#### 3.1.1.2 User Interface

We had a rough basic interface showing stats such as capacity, stamina and total earned cash. A sliding bar was added to track the amount of money collected, and a timer was used to show the remaining game time.

#### 3.1.1.3 Engine

At this point we had constructed the basic structure of the game engine, and added classes for game objects, components, transforms, drawing and loading contents. We also implemented a basic particle engine in 2D and the basic collision callback. All this work on the engine, that took most of the time, was intended to allow a modular workflow in the following phases and to increase productivity. We can now say that this approach, which mimics Unity's great architecture boosted our productivity immensely and is definitely suggested for future groups.

#### 3.1.1.4 Assets

We designed a basic game level in Unity with 3D primitives and wrote an exporter to export the constructed scene to Monogame. This allows us to use a graphical user interface to quickly prototype and position assets, and then have the same scene work instantaneously in Monogame.

#### 3.1.1.5 Effects

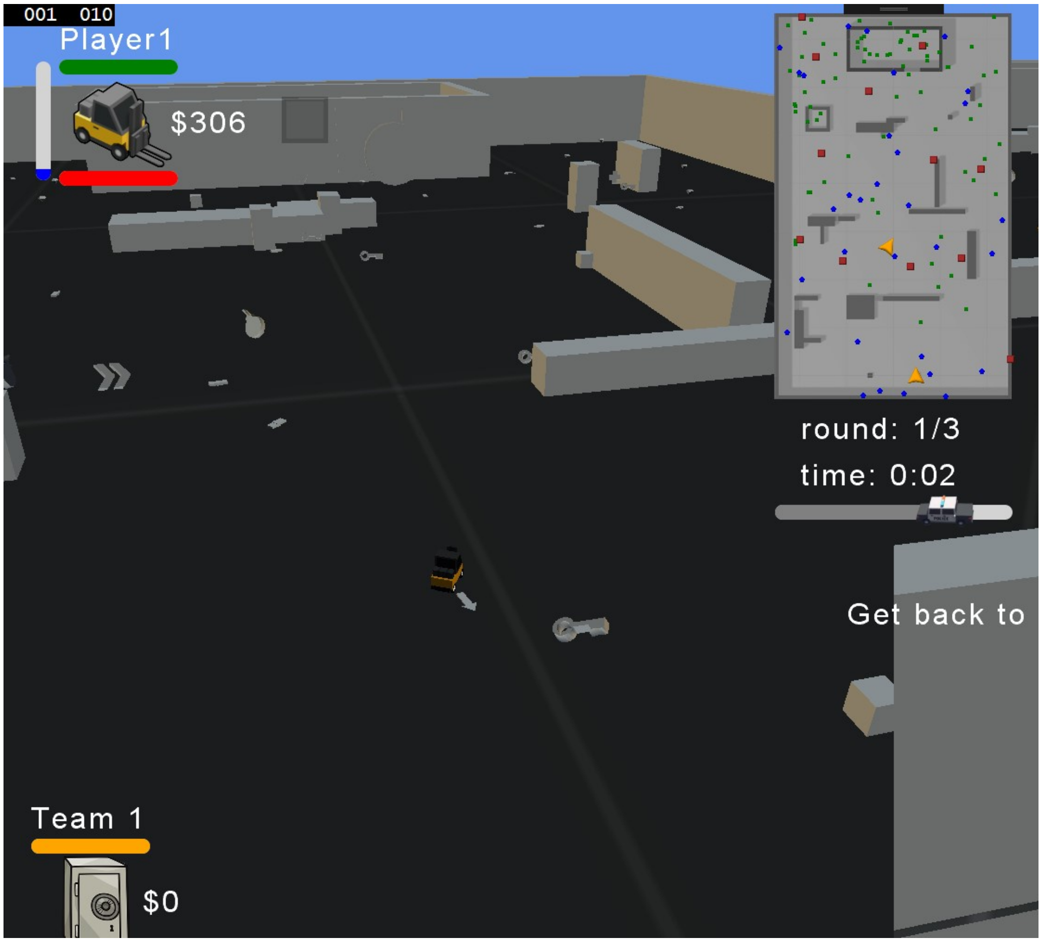
To model the final look we would like to achieve in Monogame, we experimented with lighting in Unity including light maps, moving lights and changing light colors. We also started writing shaders in Monogame for directional lights and point lights.

#### 3.1.1.6 Deployment

We managed to correctly deploy the game on Xbox at this stage.

## 3.1.2 Low Target

As scheduled, by the end of week 6 we completed our low target. This included two and four player gameplay with split screen and divided UI, attacks, death and respawn, losing coins, elaborate camera control, rigid body collisions and a more advanced unity exporter. In the screenshot below, it can be seen one half of the screen with the player in the middle, basic obstacles, a crude UI, a mini-map, timers, coins and some powerups and collectibles.



### 3.1.2.1 Gameplay

At this point, we had up to 4 users playing the game and the screen was split accordingly. Each player can rotate the camera to get a better view. Users can attack each other by performing a dash and lose money and life. In addition to unloading cash at the base, users can also choose to drop money when they are full and hence too slow. At the end of the game, we have a police coming progress bar warning users to return to the base, and if they didn't manage to do so, a cash penalty was applied to their score. We also had dynamic scene loading to change scenes on the fly (still a bit buggy, was fixed later), as well as many features from the next phase (desirable) already implemented, such as the vault with gold, openable crates, and many different powerups (six different types, such as key and bomb seen in the screenshot).

### 3.1.2.2 User Interface

We improved the UI to also show game over or restart at this point. In addition, each player has a mini-map that displays his current position and rotation, as well as that of the enemy. Important elements, such as cash, crates, powerups, vault and bases are also shown on the mini-map. We worked on custom interfaces to have the interface automatically rescale based on the resolution and aspect of the screen. This means that no matter which resolution the game is played on, the interface looks right in any case. To achieve this, we created a system that uses anchors and pivots and dynamically computes the position at which a picture or a string should be drawn such that the correct margin is kept from a particular corner of the screen, taking into consideration a pivot to allow rotation and scale. This, combined with a dynamic format to store all the UI elements, made it incredibly easier for us to adapt the UI in later phases to accommodate changes in gameplay.

### 3.1.2.3 Engine

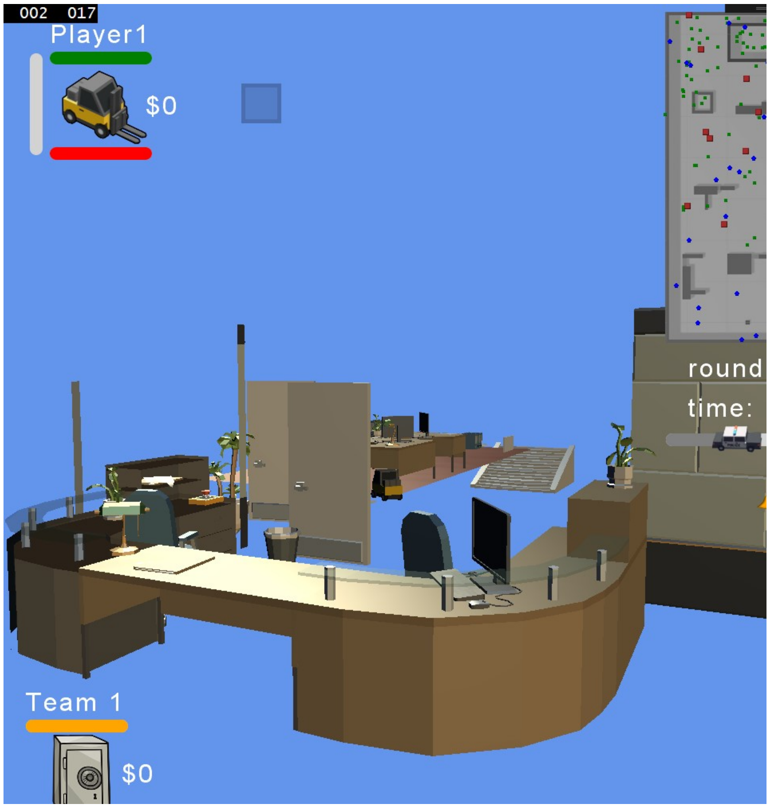
By the end of week 6, we had almost completed implementing and refactoring our game engine to ease the development in Monogame. In addition, we finished the scene loading functionality, as well as 3D positional audio. The audio volume and pitch were computed for both ears of a virtual listener such that the sound appeared to be 3D, depending on where the camera was located and pointed at. We also expanded our components to support multiple phases and to solve data racing problems. We added phases such as Awake, LateUpdate and OnDestroy.

### 3.1.2.4 Physics

At this stage, we managed to detect collisions in Monogame and have a fully working simulation, although with a few bugs to fix. We were able to simulate rigid bodies in 3D and implemented basic primitives for collision.

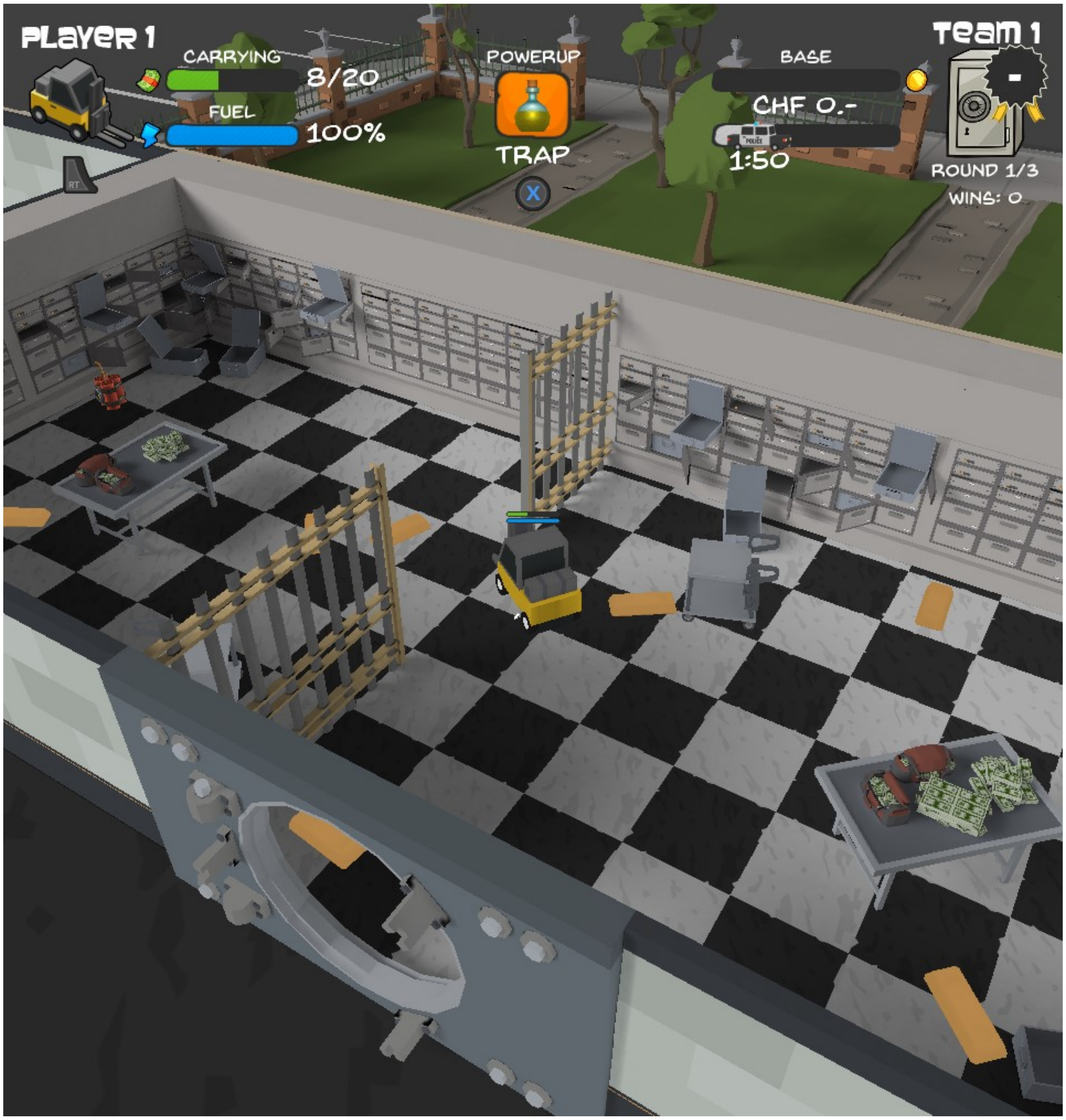
### 3.1.2.5 Assets

We continued working on asset creation, such as vehicles and powerups, and imported other assets of the bank scene in Monogame.  We enriched and expanded our level in the scene and improved the procedural spawning algorithm to generate the valuables and power-ups. A first test of importing assets with our custom importer can be seen here, which works but is not the prettiest:



## 3.1.3 Desirable Target

At the end of week 8, after other 3 weeks of work since finishing the low target on week 6 (including the Easter week) we finally concluded our desirable target. This included adding a working menu for the main screen of the game as well as a pause menu, adding all the sounds in the game, finishing the powerups (added another 5), improving once again the 2D UI to make it more intuitive and polished, adding particle effects for most of the effects in the game and most importantly, finishing working on the game scene and lighting to have a beautiful game. The easiest changes to spot are the new graphics and UI, which are showcased in the screenshot below:

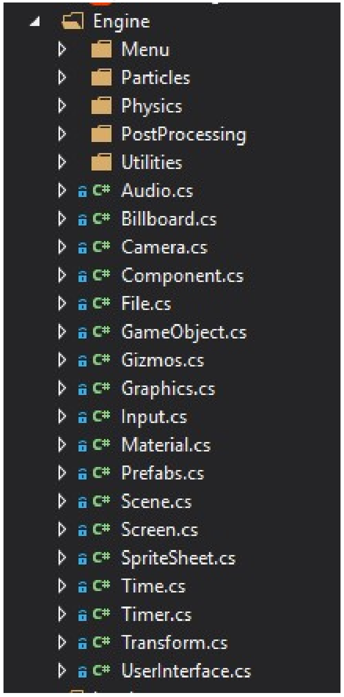


### 3.1.3.1 Gameplay

At this point, all of the features of the gameplay were completely implemented. Users can pick up and use power-ups and are able to attack crates in the scene or use a key to open the vault.  We also worked hard on balancing the game based on the feedbacks in the class and other people who tried playing the game. We aimed at playtesting the game every week with at least 5 new people (1 per group member) to keep a constant flow of criticism to improve our game, to see where the difficulties lied and to see which parts made the game fun. We worked on helping users navigate in the level, using arrows to show users which way they should go as well as the minimap, and also implemented a context-sensitive UI that provides suggestions based on what the player is doing or if he is having trouble somewhere. This means that now we didn't have to explain the game to new players, as the game was actively showing which buttons to press when it detected the user wasn't aware of an action, as well as complimenting the player when a successful action was taken or when he won a round. The robbers show up on screen with the helpful (or not) message for the player.

### 3.1.3.3 Engine

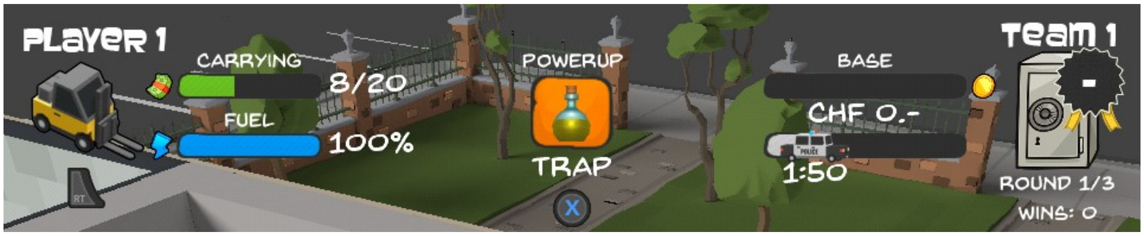
At this point, the engine is complete for all our needs and provides the following features: modular system with gameobjects, transforms and components, audio and input management, file reading and writing, graphics drawing with different shaders and materials, prefab instantiation, scene selection, time and timer management, 2d and 3d particle systems as well as 3d physics and various utilities. We are really proud of what we accomplished in this regard, as we worked hard to completely decouple the engine from the whole game, and we could reuse it to create a complete different game if we wanted.



```
Engine
    Menu
    Particles
    Physics
    PostProcessing
    Utilities
    Audio.cs
    Billboard.cs
    Camera.cs
    Component.cs
    File.cs
    GameObject.cs
    Gizmos.cs
    Graphics.cs
    Input.cs
    Material.cs
    Prefabs.cs
    Scene.cs
    Screen.cs
    SpriteSheet.cs
    Time.cs
    Timer.cs
    Transform.cs
    UserInterface.cs
```

#### 3.1.3.2 User Interface

The user interface (UI) was refined based on feedbacks (size, information shown etc.). At this stage we also have a menu working and are still working on it to make it more intuitive, nicer to look at and add selectable feature for the game such as game modes and tweakable gameplay parameters. The interface is divided between the player stats (cash and stamina), the current power-up, and team info as well as round remaining time. We also show which team is in the lead with a nice ribbon.



#### 3.1.3.4 Physics

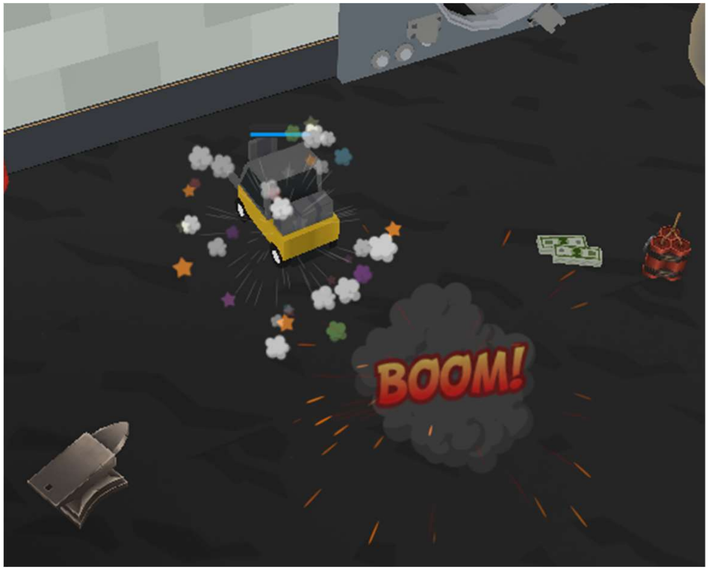
The physics engine is now also complete, and allows us to treat rigidbodies and colliders - both dynamic and static - as simple components that can be added and removed from normal gameobjects. This makes it easy to add new elements with different shapes (sphere, cylinder, cube) to the simulation without any extra coding. The engine is also optimized enough to allow us to have a number of colliders in the hundreds (we didn't test with more, but it probably also works) that is more than enough for our needs.

#### 3.1.3.5 Assets

We now have 3D assets for the power-ups, crates, vaults, and created a scene (1st level) that is fully featured. We wrote shaders that allows us to combine flat colors stored in a mesh with lighting information, to achieve a polished look with shadows and some reflections. We also created two more models: a street sweeper and a bulldozer, that the player can select at the beginning, each of which has different stats concerning speed, attack and defense.

#### 3.1.3.6 Effects

We created our sound library for the game and added sound effects at desired places, including background music, different sound effects for collision, breaks, pick-ups, menu changes, etc. We also managed to have 3D particle system in the game for most of the effects, such as bombs exploding, hits, power-ups pickup, being stunned and many more.



### 3.1.4 High Target

We started working already on many features of the high target. These are mainly additional polish to the menu and scene, such as a skybox, player name insertion in the beginning, leaderboards, loading and saving highscores (many of these already achieved), additional particles and effects for breaking crates and vaults, an improved and smarter procedural generation system that takes in a picture of the map and automatically computes a distribution function. This will allow us to have an artist paint the desired distribution of objects and the algorithm will optimize it to make it playable and fun. We also plan on adding additional art to improve the look of the game, menu transitions and post processing effects such as vignette, chromatic aberration, depth of field, color grading (all also already implemented). We plan to finish integrating all those effects in the coming 2 weeks and to focus especially on balancing of the game and polishing to have a tight, responsive and fun experience for the player.

## 3.2. Challenges and Achievements

During the development of this game, each member has encountered various challenges and problems: in this section we describe what each team member faced and their achievements toward the completion of the game.

### 3.2.1 Simone

**Simone worked mostly on creating the game engine, programming the gameplay and the AI, writing the shaders and managing the group.**

Most of my work was creating a reusable game engine such that each team member could easily work on the game without needing to know how things worked at a low level, and providing a unified interface to do so. I modeled the code in a similar manner as Unity does, with gameobjects and components. I also implemented features for simple access of input and directional audio, aligned drawing of UI and menu, 3d graphics drawing and material support with different shaders, scene management with easy loading and unloading, and transforms. Writing the engine took almost all of the first month, had many ugly bugs, especially in the math hiding behind the transform class, and I am happy I finished that. Before the physics engine was working, I also wrote a basic collision detection system that helped out testing the features.

The other big part of my work was to program the gameplay, which consisted of the following categories:
Player control: this includes driving, interaction with the scene, picking up collectibles, attacks, life and death, respawn, inventory management, stamina usage and replenishment.
Powerups: our game features a wide array of powerups, some simple such as stamina boost, some more complex such as bombs, magnets and weights. Implementing them all, with the different effects they have on the gameplay was also a big task.
Elements: other game elements, such as valuables (cash, gold, diamonds), crates, bases and traps were also part of the gameplay, and were a bit challenging to have work correctly.
Managers: managers for game modes and game states, as well as audio and prefabs.
Camera controllers: having a smooth camera follow the player without occluding the view and providing a responsive feel while also looking nice.
Shaders and materials: I had no previous experience writing shaders or materials, so that was definitely challenging, however once they worked I was pretty happy.

Challenges:
I had previous experience working in Unity, so once the hard part of writing supporting code that worked as framework for the game engine was done, I didn’t encounter too many problems programming the gameplay. I am definitely thankful for spending the time to create this similar architecture instead of working directly with what Monogame provides, because all the matrices and different ways of drawing hierarchies of 3d objects would have definitely been a big pain. Writing a simple script to act on a gameobject, and then forgetting about it and have everything work was great. The biggest challenge for me was definitely managing the group, keeping it on schedule, organizing which tasks had to be done by which week, and helping fixing something that wasn’t working. Especially being responsible for the delivery of finished features done by other members, and ensuring that everything was done correctly and fit within our theme was definitely the hardest challenges of them all. I must say I value this experience tremendously, as I believe what I learned will be of great value in my working life.

### 3.2.2 Nicolas

**Nicolas worked mostly on the level design, the tools to export the levels into Monogame, and the main menu.**

A working menu is now implemented with a tutorial section, an options section to modify the volume and other features such as sensitivity. We also have the possibility to see the high scores for each game mode which are updated after each round. In the game settings section, we can choose the game mode (2 or 4 players, 2, 3 or 5 min per round) and choose the model and the name for each player. The menu is also automatically scale for every resolution. We also have the possibility to pause the game and then restart or resume the game or go to the menu.

A fully working level is now also created instead of the crude level with only cubes. The level was designed such that fights between player is likely to happen and such that accessing the vault and the gold inside is more difficult than collecting the money around the base.

Challenges:

The main challenge was to create a nice menu with intuitive selection and display, creative features but still easily modifiable. For that we implemented an importer which parses a text file where we can easily set the constraints for the menu: the functions associated to each button, which one is currently selected and how to change between them as well as basic things like position and texture.

Designing the scene was time consuming because after the first design we realized that the position and scale of each object was important. With floating point scale and position, the result in blender (texture) was not as nice as in unity. So we had to reposition every object the best we could such that the scene is still plausible but the numbers are rounded the nearest possible to an int.

### 3.2.3 Alexander

**Alexander worked mainly with 3D particles and post-processing effects.**

I was working on the particle engine, based on a provided example from Microsoft, which turned out to be very buggy. At first I need to change a few things to get the code to compile, since it was written for an old version of XNA. After successful compiling there were still no particles visible, after a debugging session it turned out, that there were several issues with the code, as well as the communication between CPU and GPU. With a rather small Monogame community it was very challenging to get help and also in general to debug shaders.

The second issue, was also again due to old examples and tutorials, which were using DirectX 9 instead of DirectX 11. The problem was that the shader code compiled, but started to behave in an unexpected way and it was not trivial to find the cause for this behavior.

### 3.2.4 Andreas

**Andreas worked mostly on the physics engine as well as merging on git and deploying to the xbox.**

The physics part is finished. We are using the existing engine “Jitter”, which we adapted and improved to our needs. We extended the engine that it is possible to have objects which are used only for collision detection, but which are only interacting with other static objects (for the money and other valuables). This is needed so that the money which can’t be picked up is not pushed away. To integrate the physics, we wrote wrappers (components) to add to the game-objects to make the link to the physically simulated rigid-bodies. The physics-manager is finished and is used for handling the collisions between the player and any other object.

For the static objects we first tried to implement a bouncing effect as it can be seen with the bumper car. During the tests on the first levels we had it worked as we imagined it. As soon as we implemented the real level we found out that the level design is too narrow for this effect. It ended up that in some places the player got stock into a situation where the car got bounced back and forth. Right now, we are going for the solution to just project the car on the closest position, so it does not collide anymore or maybe tweak the bouncing parameters slightly.

In the beginning we have tried to simulate the car based on physical forces, which ended up into searching for the correct parameters for quite a bit. The result did not really achieve the desired effect anyway and it was tricky to control the car easily. Now we are simulating the players' car by our own calculations how it should move and simply project the physical state into the physics simulation to check and handle the collisions.

Challenges
It was a time-consuming progress to find a way to implement and integrate the physics to finally achieve the gameplay and feeling as we imagined it from the beginning. We also had to find out that for different level-designs some ideas do not work as expected (for example the bouncing effect). Another challenge was to integrate the different type of objects correctly into the physics simulation. Some are animated in our calculations and others are simulated by the physics. In the current state the integration as it is done shows us the results we had in mind.

### 3.2.5 Xingze

**Xingze worked on scripts to test out lights, importing in blender, and is now working on the trailer.**

Lighting experiments and approaches: I first experimented lighting and other effects in Unity to get an idea of possible look in Monogame including direct lights, point lights and spot lights. I also wrote scripts for dynamic lighting effects such as moving, rotating lights and changing light colors. I then tried with basic lighting effects in Monogame and implemented shaders for point and spot lights.
Sound: I created a sound library for music and sound effects in the game including background music, start and end music, effects for different vehicles, attacking, breaking, crashing, power-ups, valuables, bombs, etc. All the sound effects were converted into mono track and 'wav' formats. I also created a form for team members to vote for the most appropriate sound effects.
Models: I created other models such as the vehicles to increase the fun factor of the game. I modeled our 3rd vehicle (bulldozer) in Blender and modified a statue into Escher with cool sunglasses.
Post-processing: to have better illustration of the final look of the graphics, I tried the post-processing stack in Unity to find the effects we needed (such as color degrading, depth of field, vignette, etc.) and searched for the best parameters for each possible effects.
Trailer: we spent a lot of time working on the trailer to ensure our game can be well reckoned and promoted in the end. We first drew storyboard of the trailer, discussed how the trailer should be produced. Then we started creating the draft of the trailer in Adobe After effects, where we first cut each storyboard panel into separate layers, then place the layers in 3D and created a camera flying through the layers to achieve a 3D parallax effect. This is still in the making but we started early to have enough time.

Challenges:
One of the biggest challenges I encountered was to compile the project. Initially I tried to run Monogame on Mac OS, it then turned out that the content pipeline tool is not compatible with PC. In addition, Monogame is also not compatible with 32-bit Windows system. After spending a lot of time re-install the environment several times (including re-installing the system on my old PC and virtual machine on Mac OS), it finally turned out that besides VC distr. 2017, Monogame also requires older versions of VC distr. to compile.
Another challenge I encountered is the different coordinate system in Blender, Unity and Monogame. When constructing the scene in Unity to have a basic idea of the game, the difference in left-handed and right-handed systems making importing the assets a bit messy and additional modifications were required to resolve the problem.

## 3.3. Future Work

The future work will mostly consist of polishing and balancing the game as previously discussed. We also plan on implementing some of the goals we thought we didn’t have the time for and that were put in the Extra target. These include a website, publishing the game, and AI bots (police cars) with intelligent behavior and navigation (chasing the player, going through choke points), as well as heat-maps to further evaluate our level design and balance the game. We also allocated a lot of time to the game trailer, as well as playtesting the game with friend and refine the game based on feedback.

# CHAPTER 4. ALPHA RELEASE

## 4.1. PROGRESS

At this point, we are happy to say that we finished our *high target* and worked on most of the tasks we originally thought we couldn't complete and put therefore in the *extra target*.

For the high target, we implemented 3d particle effects, heat-maps to evaluate our level design and further balance the game, additional polish to the game and to the menu to make it more intuitive and beautiful, advanced distributions for our procedural spawning system to improve the gameplay, and added lighting effects such as flares and skybox. We also improved 2D graphics, added post-processing effects such as shockwaves for bombs that tears the screen or force-waves emitted from the magnet and additional color-grading to give a stylized look to our game.

For the extra target, we implemented AI police bots that use navmeshes and other following behavior to chase the player without colliding with obstacles, an interactive 3D menu in which the camera flies through the scene to showcase the level, single player (which we originally didn't intend but that now makes sense with high-scores and AI bots), created a website to publish our game and even got in contact with Steam to publish our game to their platform.

In this period, we also did the playtest, and based on feedback we found many small improvements, such as tweaking speeds, sizes of objects, options, distributions, colors, sounds, effects, timing, difficulty and much more. Many corrections were very small, a few a bit bigger, but after compiling a list of fixes we had around 80, almost all of which we implemented in this build.

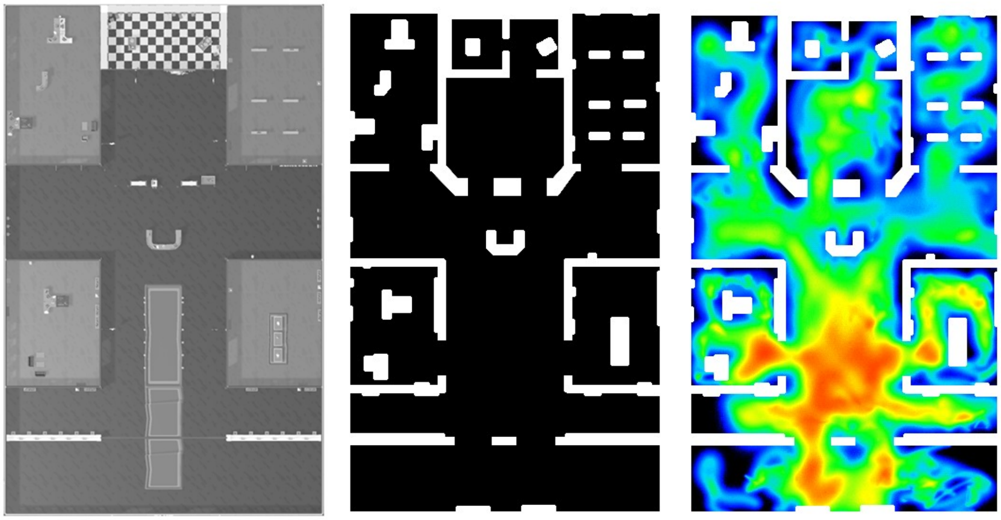
## 4.1.1 HIGH TARGET

#### 4.1.1.1 GAMEPLAY

One very important aspect of any game is balance: a potentially good game can be completely boring and unappealing if it's not well balanced. We strived to avoid that and to balance our game correctly. To do that, we worked on a good procedural spawning system that takes into account "hot areas" (frequently visited choke-points) as well as different game modes to keep the gameplay fresh.

#### HEAT-MAP

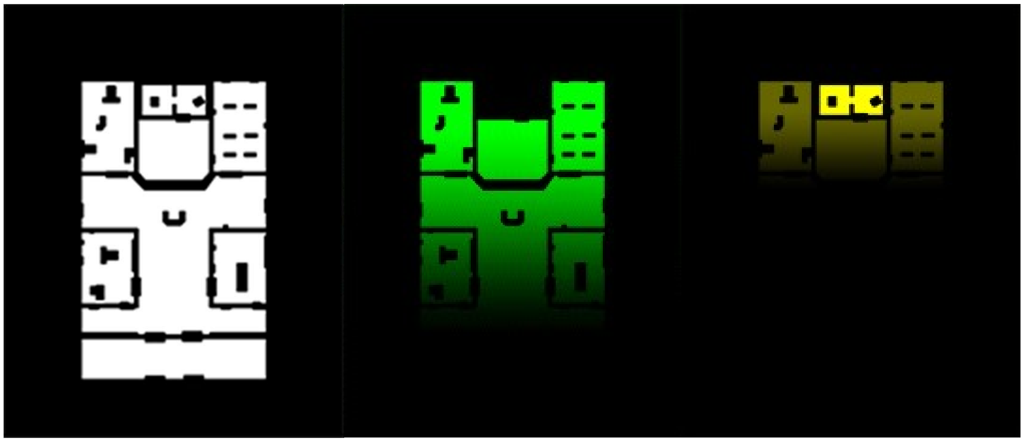
We implemented a heat-map system that computes which points in the map are the most visited and which ones are not. This helped us out to refine the obstacle placement as well as the places where to place the most valuable elements of the game to well balance risk vs reward. This system basically computes where players go, and stores this result in a heat-map after blurring it a bit to make it easier to read. We can then access this array and tint it to see movement trends. Here is a screenshot of the first bank level of our game, the accessible area and then the heat-map.



The players start at the bottom, and we can easily see that they spend most time in the lower part of the map. Also some extra action is shows for player 1 (on the left) due to the many tests in which we play alone. This showed us that the players move in a way we intended to, going less to the higher parts of the map as it’s riskier and inhabited with more police.

### Advanced spawning system

With the use of heat-maps, it was possible to spawn cash, gold and diamonds in points that inversely reflected the frequency with which players visited it. We wanted however to have some artistic control over the distribution, so we implemented a system in which an artist can simply paint in some graphic tool a desired distribution of these elements, and the system will automatically compute a distribution probability to combine with the heat-map to take into consideration both elements. For every map you can just paint with a gradient color where you would like the elements and it will work directly. Following screenshots for the cash distribution (green) and gold distribution (yellow) are shown. Note that by default we prefer valuables to be more north, and that gold has a higher preference to spawn inside the vault. Also this system avoids valuables from spawning inside walls or other unreachable areas. Following hand-painted distributions can be seen for two game elements.
Legend: white = play area, green = cash distribution, yellow = gold distribution



### Game Modes

Since we wanted to try out different parameters for our game, such as amount of valuables, crates and powerups, as well as different power-up and money distributions (meaning how frequent each power-

up or valuable is) and also multiple win conditions such as round number, duration and cash goal, we decided to create a few different game modes that the user can directly choose from the menu.
We currently have 4, explained below:

- *Default*: this is a standard game mode with all powerups enabled, as well as a fixed ratio of appearance between cash, gold and diamonds, a fair amount of crates and a timer win condition.

- *Survival*: a game mode where cash is very scarce, and that forces player to compete for the few valuables available. Only gold is enabled, with only 10 pieces available, few powerups and no crates, and a first-to-collect-5 pieces of gold win condition.

- *Super Bomber*: a mode with moderate cash in which only bombs are active, and which promotes a lot of explosive action!

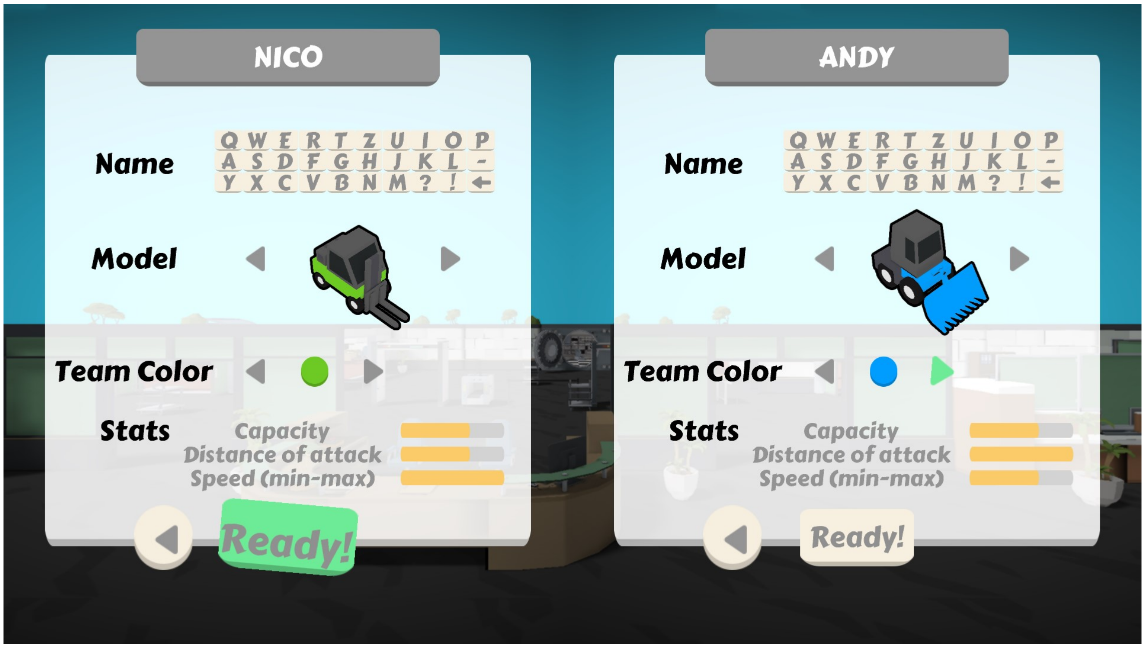
- *Crate Only*: in this mode no cash is present, except for the one present inside crackable crates. Players have to attack them to reveal the valuables inside. The exploding box power-up, which simulates a crate but explodes when attacked is very frequent, which makes it much riskier to try to open crates.

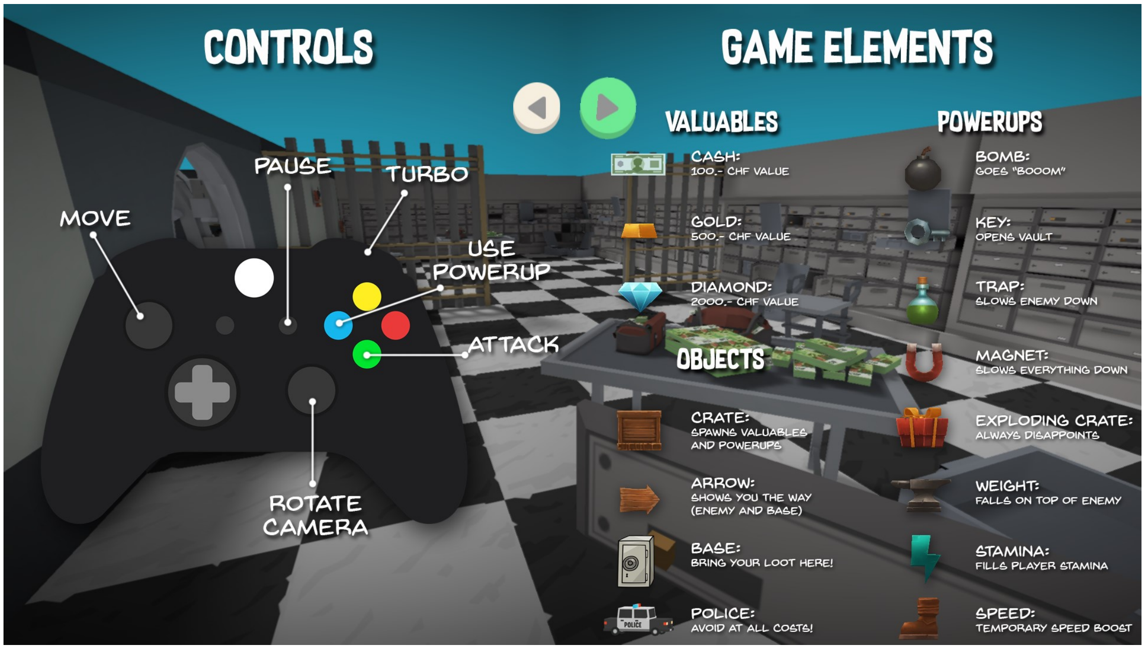
#### 4.1.1.2 User Interface

**Menu**

The menu was refined to allow the users to choose the different component we implemented. In the previous version, we had the possibility to choose the game settings: 2 or 4 players and a round duration of 2, 3 or 5 minutes. Each player could choose his name and his model (3 possible models). The menu had different panels like rankings, tutorial, options and credits as well as a pause menu. In the options, it was possible to choose the volume of sound effects.

In the new version we added two game settings: 1 player mode and a round of 10 minutes. It is now also possible to choose between 4 different game modes described above.
Each player can now also choose his team color which will is applied on the model and the base. Each model has different statistics: speed, distance of attack and capacity. To balance the game easily, each model has one good statistic and the two other are mediocre. This promotes a rock-paper-scissor dynamic in which all vehicles are balanced and no one exhibits a dominant strategy over all others. To speed up round preparation, two players can choose simultaneously their models, colors and names.
New options were also added: the possibility to have a background song or not, the possibility to choose the game difficulty (numbers of police cars) and to choose the camera control (auto or manual).
The general look of the menu was also improved by using shaders we implemented (vignette and blur) and added camera transition between the panels of the menu.




CONTROLS
MOVE
PAUSE
TURBO
USE POWERUP
ATTACK
ROTATE CAMERA
GAME ELEMENTS
VALUABLES
CASH: 100.- CHF VALUE
GOLD: 500.- CHF VALUE
DIAMOND: 2000.- CHF VALUE
OBJECTS
CRATE: SPAWNS VALUABLES AND POWERUPS
ARROW: SHOWS YOU THE WAY (ENEMY AND BASE)
BASE: BRING YOUR LOOT HERE!
POLICE: AVOID AT ALL COSTS!
POWERUPS
BOMB: GOES "BOOOM"
KEY: OPENS VAULT
TRAP: SLOWS ENEMY DOWN
MAGNET: SLOWS EVERYTHING DOWN
EXPLODING CRATE: ALWAYS DISAPPOINTS
WEIGHT: FALLS ON TOP OF ENEMY
STAMINA: FILLS PLAYER STAMINA
SPEED: TEMPORARY SPEED BOOST

#### 4.1.1.3 Engine

The engine was already completed in previous phases, so not much was added expect for additional small functions to support the current development.

#### 4.1.1.4 Physics

For the alpha release the physics got refactored and extended a little bit. It is now possible to have fully self-animated objects which are still interacting with the other rigid-bodies in the physics-world. With this extension it is possible to create the police-cars, which are following the player without the need of calculating forces. The vault uses also this technique to let the player open the vault. It was easier to calculate the transformations by our routines than calculating the needed forces to create the accurate movements. Furthermore, the calculations for the hitting test, which is used for the attack, are improved to work more accurately.

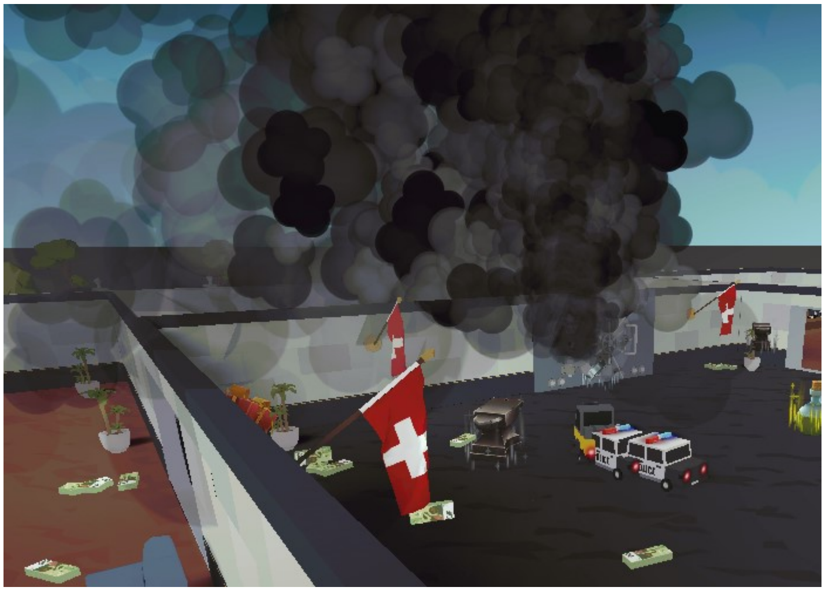
#### 4.1.1.5 Assets

Most of the assets we created in this phase were 2d pictures for particle effects. All of the 3d models were already done in the previous phases.

#### 4.1.1.6 Effects

##### 3D Particles

The 3D particle system was finished and many effects such as vault smoke, falling cash, dust and turbo as well as power-up effects were created. The big improvement over the 2D particle system is that each particle lives in world space now, resulting in more reasonable effects due to for example occlusions. With this system in place, it is relatively easy to derive new effects. Combined with projectile emitters effects like dust, cash or fire behind the vehicles could be achieved. At the beginning of the game, cash particles are flying around the level, resulting in more ambient atmosphere. One of the biggest particles effects is the comic-style smoke coming out of the vault, which is moved to the side due to a wind implementation. Another place where particle effects are used is around the power-ups given them a more highlighted view in the level. Finally, one big particle is emitted when the player returns cash to the base, resulting in visual feedback for achieving the goal of the game. This makes it fun and visually attractive to look at while playing. The big vault smoke can be seen in this screenshot:



##### Post-processing

We had a very particular look in mind for our game that we wanted to achieve. This was very stylized, colorful and cartoony. To achieve this, we decided to implement many post-processing effects to enhance the look of the game and make it stand out. The following post-processing effects were implemented and are used in the game.

- *Grayscale*: the image is turned into a grayscale image and then it slowly fades back to full color. This effect is used when a player is attacked or caught by the police.

- *Chromatic aberration*: The image channels red, green and blue are slightly shifted to get the chromatic aberration effect. This emulates the light shifting due to refraction in the lens corners and adds a bit of realism. The shift is computed relative to the distance to the center, resulting in a larger shift in the corners of the image. This effect is always turned on.

- *Vignette*: the vignette effect is used to tint the image black depending on the distance to the center. As we saw in the guest lecture about art in videogames, this helps the player focus in the center of the screen where most of the action happens.

- *Gaussian blur*: a Gaussian weighting kernel is applied to the image resulting in a blurred version of the image. This effect is used when the game is paused or the menu is visible to remove high-frequency details and to keep the focus on the menu while still showing the scene as background.

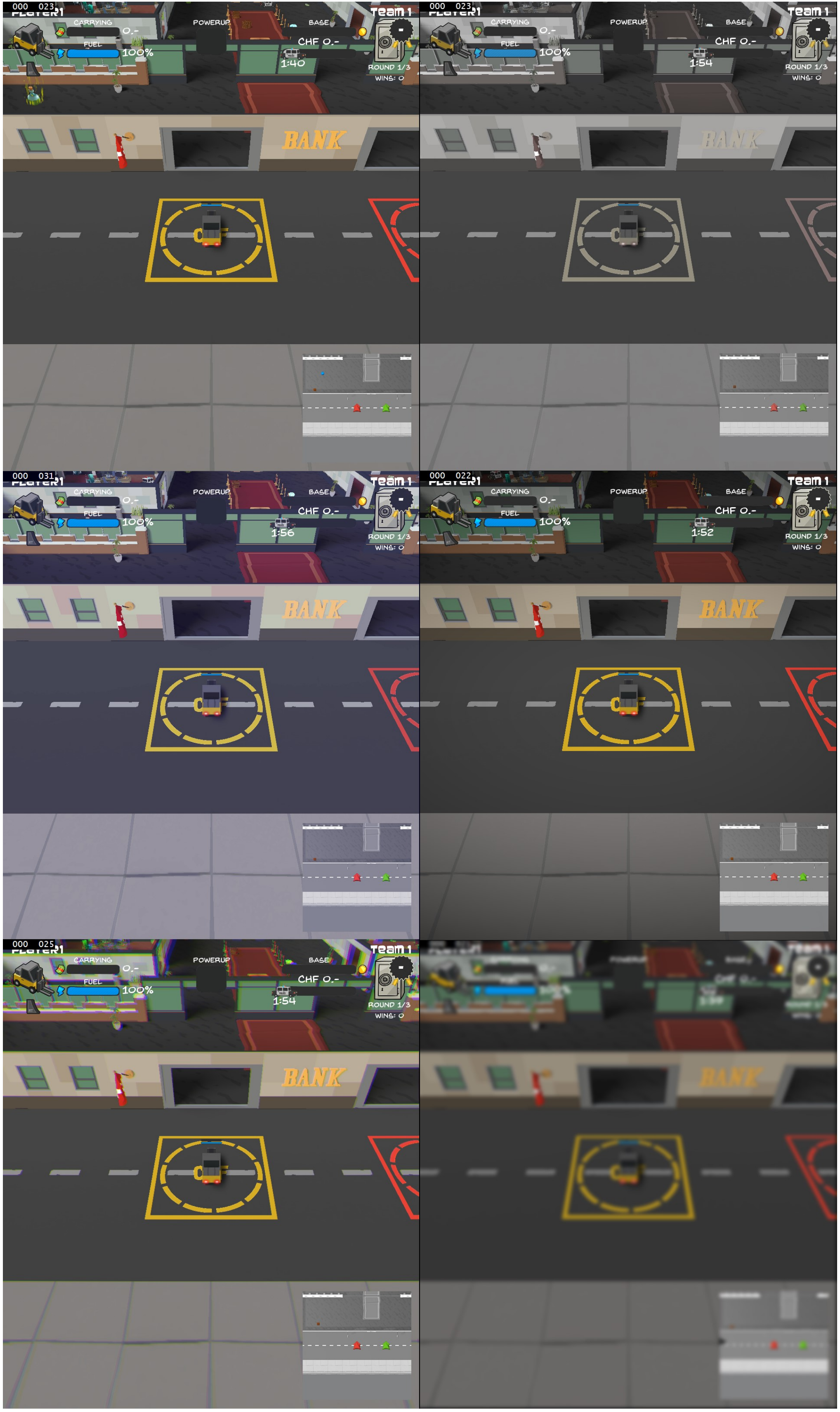
- *Color grading*: in editing software for pictures or video it is possible to change brightness, contrast, color curves and many other parameters to give a very stylized look to a frame. All these changes can be encoded in a LUT (look up table), which is basically a mapping between two RGB spaces, where to every pixel in the original image a new color is mapped based on his original color. This mapping is encoded in a small picture that stores slices of the modified RGB space. It is very convenient to be able to apply many different effects in popular editing software and then have the game apply this look effortlessly. This is one of the most-recognizable post-processing effects we implemented. With this we could add, for example, filters used in Instagram or imitate the look of popular trailers.

- *Shock wave*: the shock wave shader is used if the player plants a bomb to simulate the shock-wave that it generates. It distorts the screen in a ring pattern around the screen coordinates of the bomb explosion, displacing the pixels outwards. This shader is animated over time and disappears after a few seconds.

- *Wave*: The wave shader is used around a magnet to distort the image over time. By using a combination of sin and cos functions a black hole effect is created, which reflects the electrostatic attraction of the magnet. This is shown as long the magnet power-up is active.

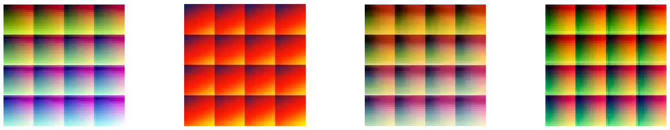
One additional enhancement was that each shader can be applied only to needed viewports. For example, when player 1 gets stun, we only apply the grayscale for his part of the screen. Additionally, we implemented that the parameters can be specified for each viewport differently, as for example the shockwave center is different for each player.

Most of the effects can be seen in the picture below. Top left to bottom right, row-by-row: normal, grayscale, color grading, vignette, chromatic aberration, blur.



### LUT

Many LUT pictures to be used in the game were created. Each of them encodes a particular color filter. Two approaches were used for generating good-looking LUTs: the first approach we tried was in Unity with the post-processing stack plug-in, where we applied different adjustments such as color grading to the scene and stored the results to a neutral LUT picture with grid size 4x4. We also tried generating LUT pictures using Photoshop with its provided filters. We wrote a MatLab script to cut the LUT pictures in the required grid and pixel size, and applied the filters to a neutral LUT picture. The LUT pictures are then used in the game to add different flavors to the game. In total we have about 40 LUT pictures for different effects and moods. We then selected the most appealing ones and added the possibility for players to pick the one they prefer in the game. Some of these LUTs are shown here, starting with the neutral one and showcasing a few stylized ones:

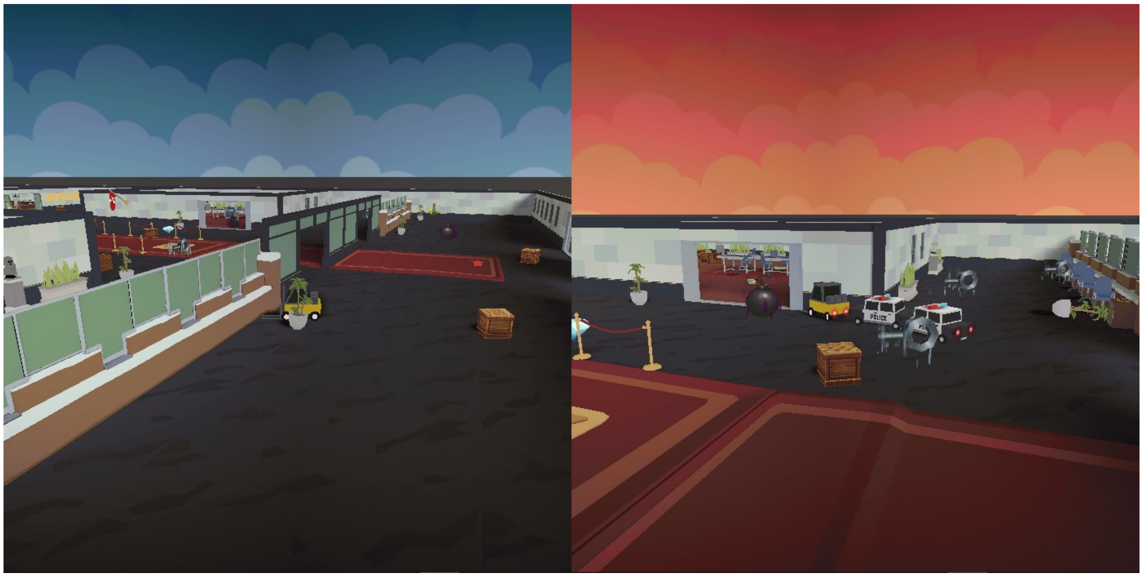


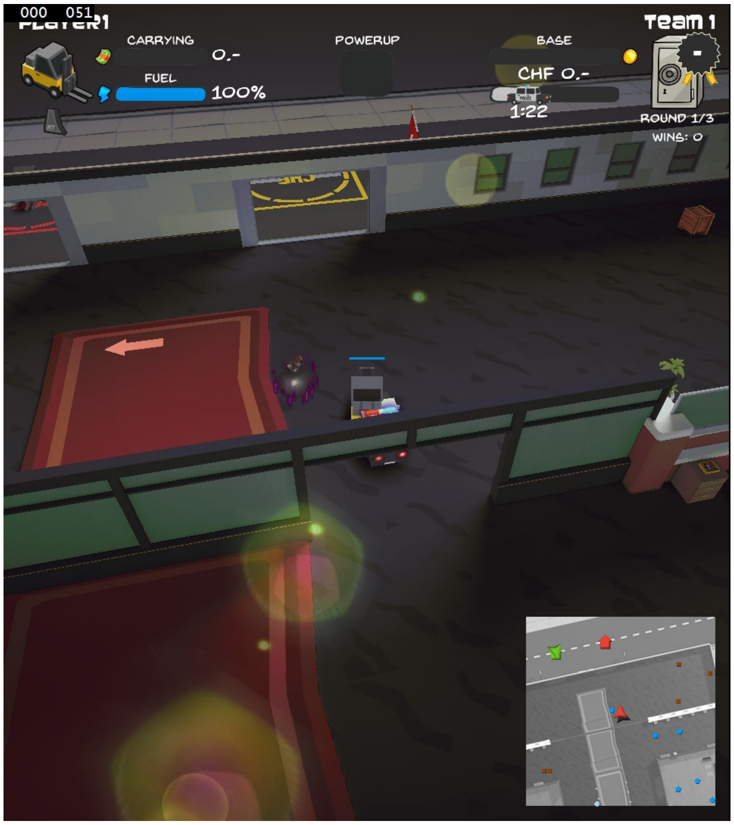
### Additional Polish

As a last visual improvement, dynamic shadows were implemented for the different moving objects such as players, police-cars and dynamic objects. It now looks more realistic as all the objects cast a shadow. Additionally, we are simulating the blinking for the police lights with a simple sprite-animation. With the same technique we simulate the back and front lights of the cars. Finally, the wheels of the cars can be rotated so that they rotate into the correct direction when the cars are driving along a curved path.

### Skybox and Flare

We wanted to add a touch to the sky as well, so we implemented different skybox effects for different times of the day, and a sun that creates a flare when the player looks directly at it. Currently we have skybox textures for daytime, sunset, midnight and daybreak, and these textures are loaded randomly to make the game more dynamic. Two skyboxes are shown here, as well as the flare effect:





#### 4.1.1.7 Deployment

Deployment continued without problems to the Xbox with all the features working.

### 4.1.2 Extra Target

Originally, for our extra target we wrote the we would like to have a single player mode with bots controlled by the computer that can navigate the level intelligently to provide a challenge, as well as more powerups, persistent state between rounds, and publish a website documenting our game with links to download and possibly an external store. In this section we document what we achieved of these tasks.

#### Police AI with navigation

The biggest challenge of the extra target was to implement bots that could drive realistically through the level without colliding with obstacles, exhibiting somewhat intelligent or plausible behavior, and without ruining the balance of the game or becoming frustrating. For our game, we thought that police car bots were well suited and fitted the theme. Based on the difficulty selected by the user at the beginning of the game, a number of police cars will be spawned during the curse of the round. Some of them navigate independently in the level protecting high-risk areas such as the jewelry and the entrance of the vault, whereas some others actively try to chase the player. Both types of police avoid obstacles in the level. As the rounds slowly comes towards the end, the police will accelerate to pressure the players into going back to the base and pay more attention to not get caught. After playtesting, we found that it was more fun being able to also attack the police to briefly stun them and be able to escape, so this is a feature that is now in the game.

### Other powerups

We had previously already implemented more powerups, such as the exploding box, the oil trap, the speed pad, the falling weight and the magnet. In this phase their parameters were tweaked and balanced to make them more fun. We also got some cool power-up ideas from feedback from the playtest that we might try to implement.

### Persistent rounds

The game now features the possibility of playing multiple rounds, with stats such as number of wins and total cash earned stored persistently to decide the global winner at the end. Persistent leaderboards are also saved locally to allow friends to compare who is the best “Failed bank robber”.

### Website

We wanted to have a website for our game for people who would like to check out our game, with screenshots, link to download, gameplay video and various information about the game. Although the game website is not finished yet and some of the links (for example to download and to the various stores) still don’t work, the first draft of the website can be seen here:

www.thatfailedbankrobbery.altervista.org

### Store

We started the process of publishing our game on the Steam platform. We never did this so we assume this will take a bit of time, however the process is going smoothly for now. We think we will be able to get the game published by the end of the semester.

## 4.2. Future Work

Except for a few other tweaks that should be finished by the end of this week (week 11) we can say we are done with our game. We will continue playtesting our game and polishing features and refactoring code in the remaining 3 weeks, trying to balance the game as best as we can to continue improving it and make it more fun. We are also currently working on a second full level that will be done by next week, with a labyrinth-style layout map that will encourage a different type of gameplay. We will be working on the trailer for the final presentation, publish the game once we have a gold version as well as try to showcase it a bit in the gaming community.

# Chapter 5. Playtest

## 5.1. Playtesting Session

At the start of May we organized a formal playtest. By that time, we had accomplished all of the high targets and most of the extra targets. We were seeking for get ideas and feedback to balance the gameplay, improve the general look of the game and tweak the effects for a better game experience.

### 5.1.1 Participants

Each member of the team invited about 10 to 20 people not in the class to the playtest session including classmates, friends and family. During the playtest session we had around 60 participants in total that tested our game, out of which 38 filled the feedback form. Our participants varied in gender, age, nationality and experience in video games. The diversity enabled us to get a more rounded view of the game.

### 5.1.2 Organization

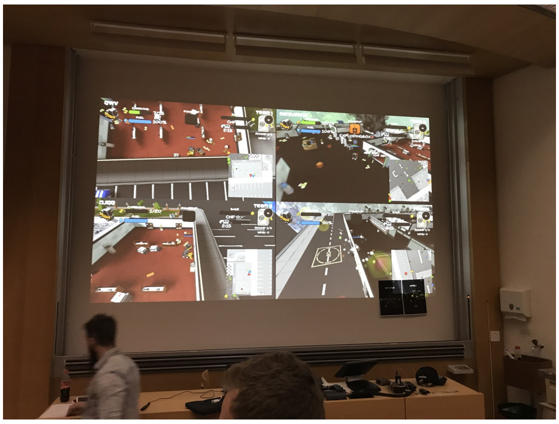
We booked a big lecture room for 4 hours to perform the playtesting with an Xbox and two laptops for playing to allow a maximum of 8 concurrent players at all times. We also had two laptops for filling the feedback forms and an ipad for players who would like to be credited to fill in their name. The Xbox used a projector and supported up to 4 participants playing at the same time. Participants could also play 1 vs 1 on the laptops. We had in total 8 controllers that supported multiple people to play at the same time on different machines to reduce the waiting time.

When a participant or a group of participants came, we first welcomed them and provided them with refreshments. Then we allocated the participants to one of the 3 playing stations if there was one available. Participants were also welcomed to watch the competitions between other players during the waiting time. During gameplay, each member of the team took care of the guests he/she invited, observed and recorded the opinions the participants had during the play and answered questions. After a participant finishes playing, we then asked them to fill the questionnaire regarding the game experience or leave any suggestions or feedbacks vocally.

### 5.1.3 Live from the Spot

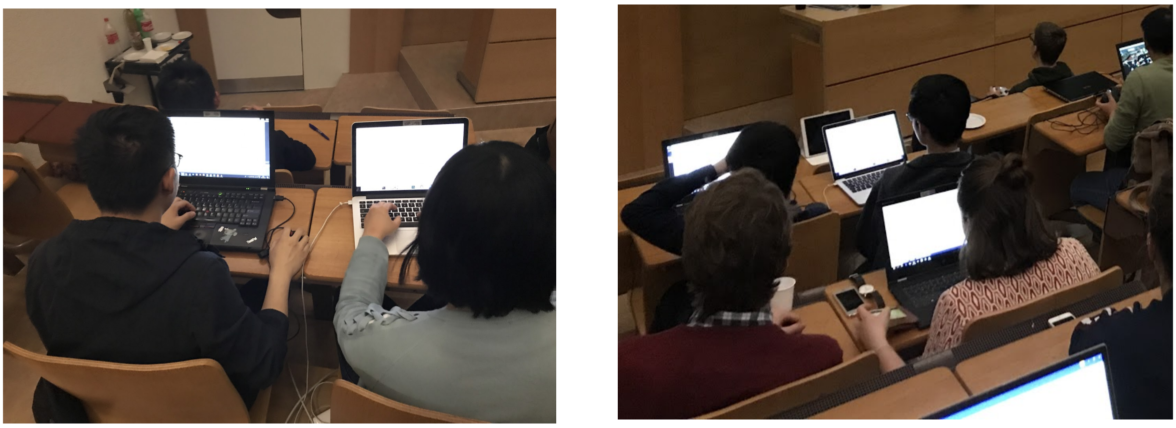
Our playtest went very smooth. By leveraging multiple machines for playing and feedback, most of our participants didn't spend time waiting. We got the chance to test our game on Xbox for different number of players, and received a large amount of feedback for the game experience.

- **Play on different machines**



- **Participants in intense competition**

- **Participants filling feedback forms**



## 5.2. Questions and Comments

### 5.2.1 Questionnaire

Our questionnaire was designed with the following sections:

#### 5.2.1.1 Overall Feeling

We wanted to know what the players though about the 3 game pillars we designed our experience around, namely that the game should be fun, simple and beautiful. We asked the participants to rate from 1 to 6 for these aspects:

- Was it fun to play the game?
- Was it simple to understand the goal?
- Was it easy to control the game?
- How was the general look of the game?

#### 5.2.1.2 Gameplay

This section was aimed to test if our gameplay is well-balanced and seek solutions for balancing game settings. We designed questions regarding parameters of game objects (vehicles, valuables, power-ups) and functions (camera rotation, minimaps, etc), and asked the participants to rate from 1 to 6 or select the best fit for the following questions, as well as which one was their most and least favorite and if they had any additional feedback:

- **Gameobjects**
  - **Vehicle**: size, speed
  - **Valuable**: size, amount, setting
  - **Power-up**: size, amount, setting

- **Functions**
  - **Map:** size
  - **Minimap:** size, helpfulness, rotated or fixed
  - **Camera rotation**: rotated or fixed
  - **Arrow:** size, helpfulness

#### 5.2.1.3 Graphics

This section was aimed to test the general look and the special effects used in the game. We asked the participants to rate from 1 to 6 for the following:

- Brightness/filter used
- Appearance of the models
- Lighting effects
- Special effects such as bomb explosion, smoke, etc

#### 5.2.1.4 User Interface

This section is aimed to test if the UI (menu, tutorial, instructions, user statistics during the game, context sensitive UI) is user-friendly and seek solutions that improve the look and usefulness of the UI. We asked the participants to rate the following:

- Size of the UI
- How intuitive the UI is
- Intelligence of the context sensitive UI

#### 5.2.1.5 Sound

This section was aimed to test the sound effects in the game and to improve the sound and music to provide a more dynamic game experience. We asked the participants the following questions:

- Volume balance between sound effects and background music
- 3D sound effects
- Selection of background music

### 5.2.1.6 Physics

In this section we wanted to test the physics used in the game and find suggestions for more realistic and immersive mechanics. We asked the participants following questions:

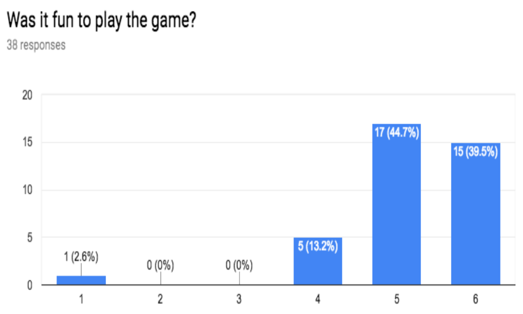
- How realistic are the physics?
- What did you like/dislike?
- What could be improved about the physics and control?

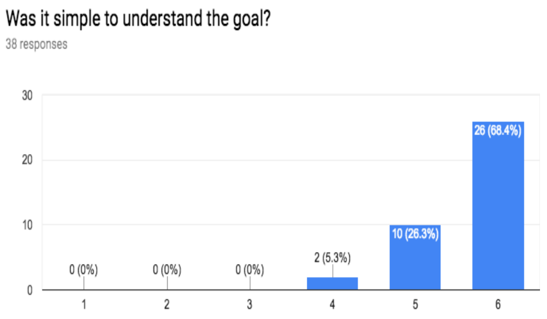
### 5.2.1.7 Open Questions

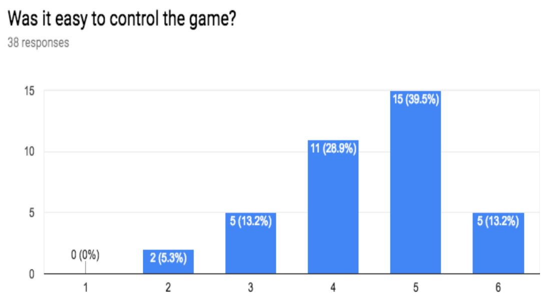
For each section, users are welcomed to leave additional/specific feedbacks or their suggestions and opinions towards some particular element in the game. We often asked both what they liked and enjoyed and what was annoying and could be removed to find both strengths and weaknesses of our game. We also asked in some sections if they had further ideas to expand the game, such as additional power-ups or effects they would like to see implemented.

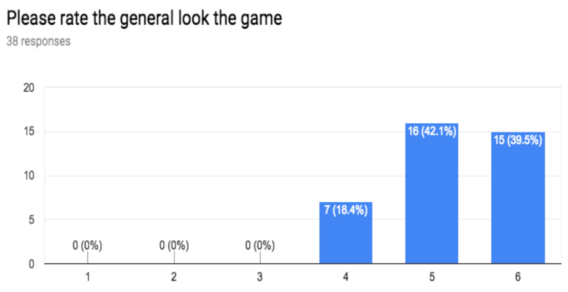
### 5.2.2 Feedback

We have gained positive feedbacks in many aspects. Most people think our game is fun to play, easy to pick up and appealing to look at. In the following distribution we can see the different distributions. Since some users reported that the game could be made easier to control, we focused a lot of work in the following weeks on solving that problem.







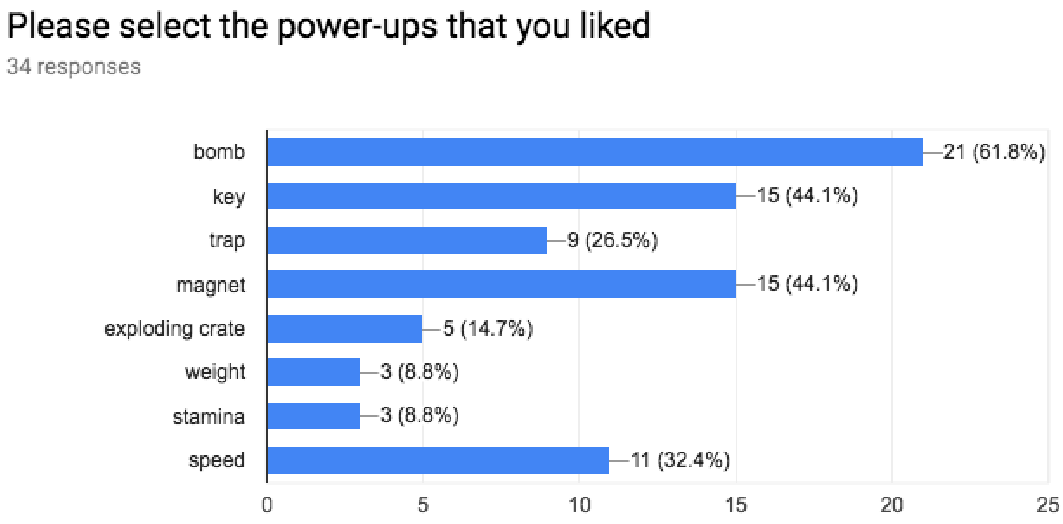


In each section, we also got many helpful feedback and some advice:

### 5.2.2.1 Gameplay

- **Gameobjects**

  Most participants are satisfied with the settings of the game objects such as size, amount and speed, though some of them think gold and diamonds are a bit fewer than expected. Our participants like the different types of the power-ups, and selected their most and least favourite ones. The most favorite was the bomb and the magnet, the least one the key. This information is very helpful for us to modify and improve our design of the game objects and tweak their different distributions.



Please select the power-ups that you liked

34 responses

  We also gained advice and ideas on additional power-ups such as making vehicles fly, making police chase the other player, fake gold to fill the opponent’s cart, and disrupt the opponent making him move in the opposite directions. These suggestions provided us with more rounded outlook of the features that may be implemented in the next release or future work.

- **Functions**

  Most people are satisfied with the functions provided in our gameplay such as the minimaps and camera rotation. The feedback helped us with some design decisions particularly. For example, we were hesitating on whether to enable camera rotation or make the camera follow the vehicle automatically. The survey suggested that people prefer automatic follow and fixed minimaps, though some of them didn’t notice the functions when they are focused on playing.

### 5.2.2.2 Graphics

Most people are satisfied with the graphics effects implemented and the models used in the game. We got very constructive feedbacks about some disturbing or less perfect parts of the look in the game. Many participants complained about the smoke in the vault area and the colors of some barriers and the ground. These dissatisfactions were changed easily and were essential for us to improve the game experience.

#### 5.2.2.2 User Interface

Most people are satisfied with the general feeling, size and intuitiveness of the user interface. From the feedback, we also got suggestions on improving the UI such as making the input keyboard bigger, interactive/video tutorial and more user statistics at the end of the game..

#### 5.2.2.2 Sound

Most people are satisfied with the background music and sound effects. Many of them didn't notice the 3D sound effects (this may due to the noise in the playtest session as everyone was talking and we didn't provide headsets).

#### 5.2.2.2 Physics

Most people are satisfied with the believability of physics implemented in the game. We also discovered some bugs during the playtest (which is good that they are found before the alpha release). Several people reported that they got stuck in some corners, or sometimes the vehicles overlapped with each other when attacking. Some people even got outside the map during the game. Bugs found at this stage helped us to ensure the quality of the game as we can fix them before the final release.

## 5.3. Design Revisions

Based on the feedbacks, we reviewed our design of the game and decided to tweak some of the settings and fix the bugs found during the playtest. We compiled a list of different bugs and fixes, suggestions and improvement, and got over 120 in total. We split the work and implemented all the fixes in the next weeks.

### 5.3.1 Gameplay

To make players with various video game experience enjoy the game more, we decided on adding different game modes (different number of police cars, various valuable ratio, different number of bombs, etc). We also adjusted some of the settings of the power-ups, and implemented some new features suggested by the participants.

### 5.3.2 Graphics

Our changes regarding graphics were mainly on solving the dissatisfactions from the users, such as putting less smoke in the vault area and changing colors on some of the game objects to make them look more distinguishable. Placement of barriers and other objects were also rearranged to improve the experience.

### 5.3.3 User Interface

From the playtest we found that our user interface could be further improved, some key elements were often ignored as people concentrated on the gameplay. Our changes for this section is mainly on balancing the current UI elements and tweaking positions and size of some icons.

### 5.3.4 Sound

We decided to make the background music looping in each round, and randomly load a different song for each round so that people don't get bored.

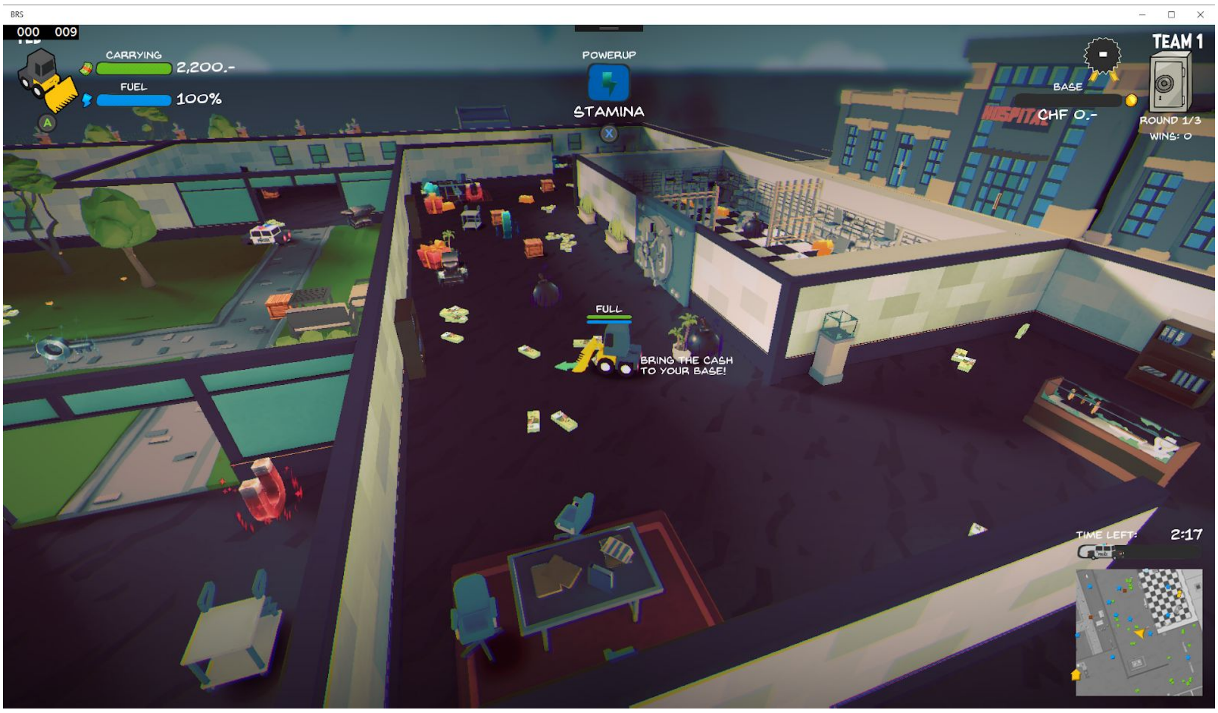
#### 5.3.5 Physics

Our changes on this section were mostly fixing the bugs reported during the gameplay. We also decided to add some new features such controller vibration during collision and rotating the wheels as suggested in the feedbacks.

In conclusion, we can say that the playtest was a great experience. It was the first time we had so many people come play our game and it was really beautiful to see people having fun while trying to steal the most cash from the bank. Most of the fun came out of the competitive and cooperative aspects of our game, as well as the interaction with the different game elements and the police. We can say we are happy that people enjoyed the game and take this occasion to thank everybody that showed up to play our game and give helpful feedback. Thank you all!

# Chapter 6. Conclusion

## 6.1. Final Results



*Compared with our alpha release, we have improved our game look by adopting different LUT files, with more features and fixes added based on feedbacks from the playtest.*

## 6.2. Experience

*Were you able to follow your development schedule, or did you deviate significantly from it?*

*We were able to follow our development schedule tightly. We followed an agile development cycle with well defined weekly tasks specifying what each member had to do, and had weekly meetings to monitor what was achieved and what was still missing to stay on track. The great amount of time invested in planning and making sure that everyone had tasks that fit their strengths allowed us to complete all tasks that we wanted to achieve in the final game.*

*How did the different elements of the project structure (development schedule, prototype, playtesting, etc.) contribute to or hinder your progress?*

*We think that is was very good to have a structure with functional, low, desirable, high targets to achieve as this helped us split our game into well defined products at the end of each development cycle. However, the suggested development schedule is much behind the actual time needed for the game. Considering the time to get accustomed to and learn the Monogame framework, it is too late to only present it until week 4.*

*The physical prototype helped us a bit as we found out things that could be improved (obstacles, placements, etc) when playing on the paper, but it also takes a lot of time to construct and is not representative of the possibilities of the computer.*

*Playtesting wasl a helpful way for feedback in order to keep our development in the right direction. But the schedule of the playtest was also too late, as a large amount of time was required to implement all of the feedback into the game.*

*Do you feel there wasn't enough time or that the schedule was too compressed?*

*We think the course goal to create a cool game in 3 months is an achievable task if each team member knows how much time is required and the available time is planned and used in the most efficient way. However, we felt following the proposed schedule might be compressed to finish all the desired features on time as we started development much earlier than the proposed schedule.*

*What was the biggest technical difficulty during the project?*

*One of the biggest technical difficulty during the development was not with the game itself, but rather with the Monogame framework and Xbox. We encountered intrinsic bugs from the framework, uncompilable demos, things working on laptops failed on the Xbox. We spent a lot of time on finding a good solid physics framework which is also working on the xBox. Within the same it was then easy to simulate the overall environment, but rather hard to achieve the handling of the vehicles which are not physically simulated. To handle this interaction correctly was challenging and still not perfectly done.*

*Another was to rewrite everything to be able to use the hardware-instancing in late state of the project (2nd last week). It was late and the testing was rather poor at this stage as it was also coupled with the multithreading issues we occured only at the end of the project.*

*Managing all of the team members efficiently is another "technical difficulty" we encountered. Making sure everybody followed their personal schedule, gave the required results on time, and also usually finding ways to solve what others didn't manage to do in the allotted time was definitely not easy.*

*Do you consider the project a success?*

*We think our project is rather successful as we achieved all our targets and even the extra targets. In addition, we could have probably achieved even more when we had worked more together in the same room to build the project together and not everybody his part alone*

*To what extent did you meet your project plan and milestones (not at all, partly, mostly, always)?*

*We met all of the project milestones. We had specific weeks to achieve each of them (eg. desirable target by week 8), and all of them with all of the features were always reached in time (except desirable that required 3 more days). It was also good to have planned for a 3 weeks buffer at the end since most of the unexpected work came up in this period.*

### 6.2. Personal Impressions

*How well did your initial design ideas materialize into the final game?*

*We think the final game is representative of the initial ideas brainstormed at the beginning. Though we added new features that came up during development and removed some that didn't enhance the gameplay after refactoring, the core mechanics of the final result fits our initial design and met our 3 guidelines: simple, fun, beautiful.*

*Did the course meet your expectations?*

*The course has provided us a great experience in creating a game from scratch and exceeded our expectations in a lot of ways. We were challenged to deepen our knowledge in various subjects and got the opportunity to create a game as not only programmers, but also designers, artists, producers and managers.*

*It will be great if more inputs and guidelines can be provided in the earlier state of the course. Some interesting concepts about camera, lightning etc would have been better placed in the earlier schedules instead of the last week. Also some common problems which might occur (performance drain => hardware instancing) would be helpful to be quickly reviewed in a lecture.*

*Are you happy and proud of your game?*

*Overall we are very proud of our game. We managed to create a fun, simple and beautiful game with a whole team working together. Although there are still some possible improvements, we think our game is already in a polished and stable state. Seeing people play our game while having fun was definitely what made us the most proud about our finished product.*

*What was your impression of working with the theme?*

*The theme can be interpreted in many different ways and allows a lot of possibilities. In addition, we think that the course staff should enforce more interpretation of the theme, and a theme that can be visualized better may be used in the future. We as well other groups all simply integrated the theme within the story. But looking at the game nobody would ever make a link to the theme itself.*

*Do you think the theme enhanced your game, or would you have been happier with total freedom?*

*We agree with the idea of having a given theme makes people more creative by having boundaries. However, with this theme we felt that it is still possible to do anything freely by linking a story to it. In the beginning it was hard to find something that both links the theme and interests us, but once we settled on the bank robbery we were really satisfied.*

*What would you do differently in your next game project?*

*Probably we would spend more time on core mechanics and balancing the game. Looking at it now we felt that we settled too early on the game mechanics without questioning them anymore or testing different variants, and didn't look much at what could have been changed to make the game even better from the beginning.*

*In addition, we would spend more time and investigation about building a solid framework as the amount of refactorings in the game is overwhelming. We would definitely spend more time thinking about performance and code safety during development and pay more attention for the code reusability. If it's possible, we would use a framework which is currently more active.*

*What was your greatest success during the project?*

*We are very glad that we have met the initial ideas (fun, simple, beautiful), implemented all feature we initially wanted and seen people having fun playing it.*

*We think our game engine is rather successful as it really simplified everybody's work and allowed us to achieve more. We are also very proud of our 3D physics engine, particle effects and the visual look of the game.*

*Are you happy with the final result of your project?*

*Yes, we think the final visual result is very pleasing and we managed to create a game at this scale is really rewarding.*

*What improvements would you suggest for the course organization? (perhaps in D1 evaluation)?*

*We really liked having a professional from the field as a mentor and we have gained a lot of feedbacks and advice from him, but it would be better if we can also meet our mentors face to face instead of only online chatting.*

*We think the course staff should follow more closely the development of the games. It started really well with everyone giving good feedback on the project description during the first few weeks, but after that there was almost no feedback, especially from the technical side, such as solving performance issues in monogame with instantiating, etc.*

*It would be great if more transparency on how the games are judged (and how much time they spend on trying the games) and maybe having 3 places for the 3 best games not only the first.*

*In addition, if more motivation can be given during the course would be great. The 10 unsupervised credits really are a temptation for free-riders. We could clearly see the difference between motivated people who put much more effort than required, for which 10 credits definitely aren't enough (even though they are not the motivation), and people who basically got 10 credits with too little effort. People know that if you have something that moves on screen you get a pass grade and thus may not be highly motivated. Also just getting a pass or fail grade regardless of the final result and effort is also kind of a pity.*

*Did you like using MonoGame?*

*We found monogame a bit buggy and outdated where some of us encountered loads of compilation issues at the first few weeks due to different operating systems, visual studio and VS plug-in versions. Although spending time investigating invent the wheels can be interesting, it is also handy to use a more finished framework where we can focused on the more advanced aspects.*

*Based on previous experience, we think Unity would be a more appropriate choice as it wouldn't give such a big incentive to people to go 2D and therefore avoid all the complex 3D stuff, and would allow us to focus on more interesting aspects rather than trying to import a video, an animation or managing content. It was definitely cool to learn a new framework and code on top of it, but also the lack of user-base and online resources was something to keep in mind.*