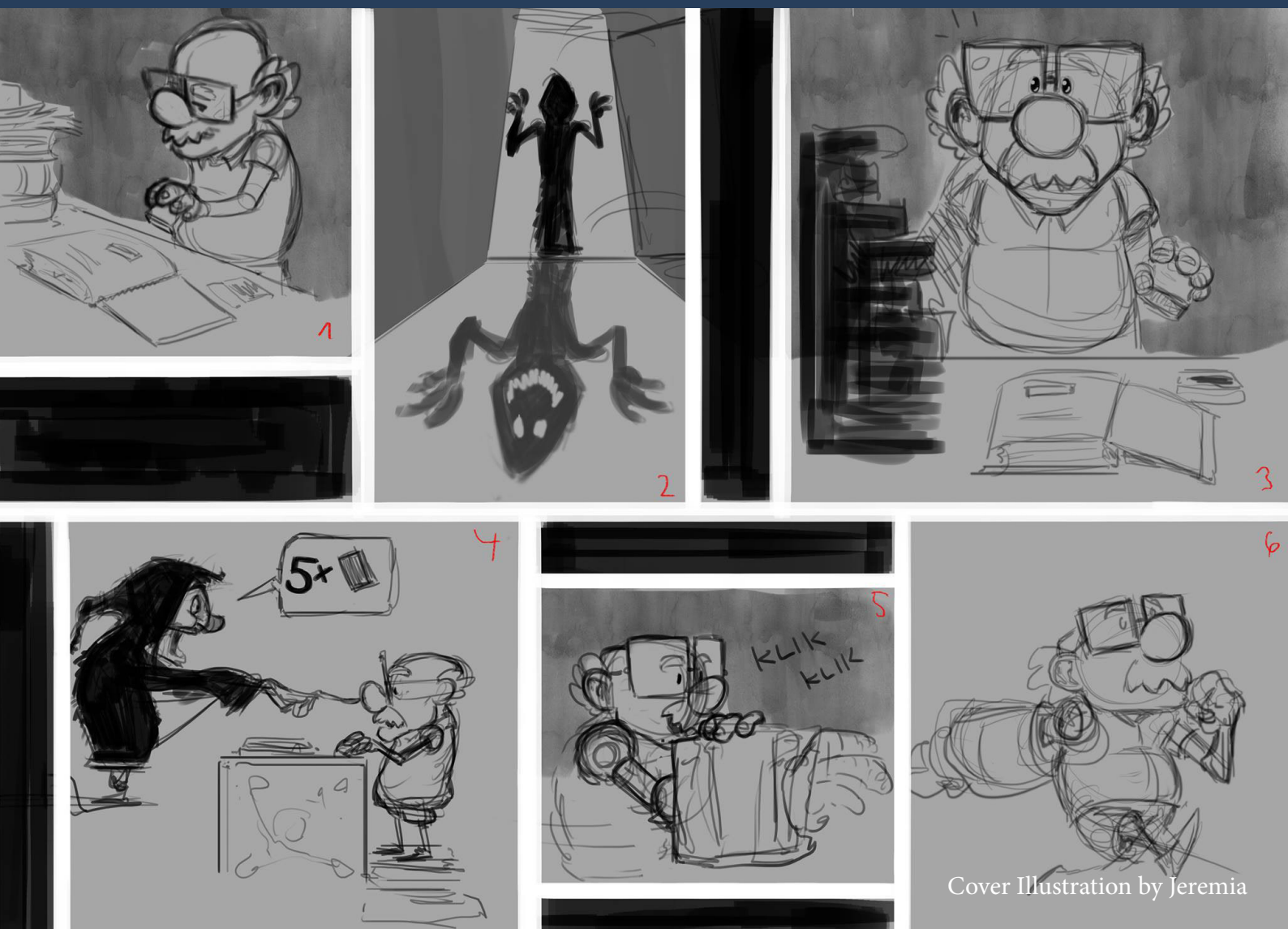
ARMVILLE

Archie's Book Chase

REFLECTIVE PRACTICE



Cover Illustration by Jeremia

ARMVILLE - ARCHIE'S BOOK CHASE

REFLECTIVE PRACTICE

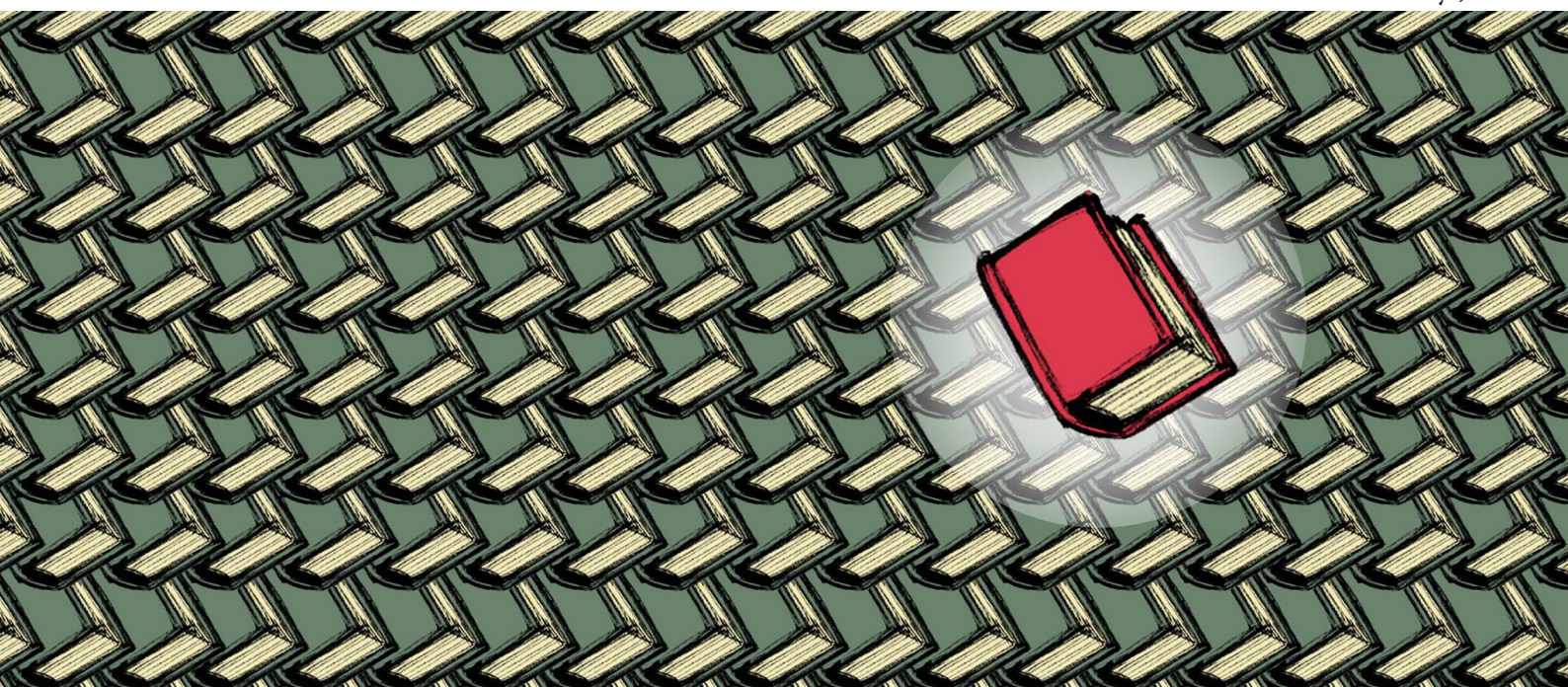
09/09/2014

Hien Quy Tran
Student Number: 543800

Lecture: Game Design
3rd Semester
Project: Development of a 2D Side-Scroller in Unity

Content

Illustration by Jeremia



Concept

"Armville - Archie's Book Chase" is a nonlinear, high score based time trial runner. The core idea was to create a game with a high replay value. Also, we wanted a game, which provides a variable challenge according to the skill and intention of the player. While the first trial could be challenged by getting to know the controls and the puzzle like variations of combining the game's mechanics with the definite goal to reach the end of the game, the second attempt could already be driven by the eagerness of improving the learned controls, ultimately resulting in the desire of finding the best route and beating the current high score. In order to improve the replayability of the game, we decided to design a nonlinear level with multiple alternative paths. One of our primary goals, was to avoid the existence of one particular path which would have an advantage over all other paths. Since now, we have found three different paths to be equally fast, compared to all others. The focus however was addressed towards the unorthodox/unusual game mechanics in the movement of the main character.

Mechanic

Unlike most games, in which the character is performing a vertical jump by pressing a single button, our character is lacking the ability to jump in the conventional way. Instead, it is substituted by a free rotatable arm, which he can use to punch onto solid ground in order to gain repulsive force for rebound. The position of his right arm is directly translated from the position of the controller's right stick. Giving the player freedom to rotate the right arm of the character in 360 degrees, independently from the characters body, to any given time, allows the character to punch onto any solid obstacle, including walls and ceilings in order to gain repulsive force for rebound into opposite direction as the punch was executed. This enables a high variety for individual precise jump and wall jump combinations. Adding a feature, we call "the grab", to the mechanics of the free rotatable arm, extends the variety of how the player is able to progress through the environment tremendously. Not only is the character able to use his arm for gaining repulsive force for rebounds, but also to use his hands for grabbing corners of obstacles. This enables him to pull himself up the edges or swing around corners.

Art Style

To provide a nonlinear level design, we decided to install checkpoints all around the level, which the player has to reach in an unrestricted order. We decided on a library as a setting, which provides a believable reason to reach certain checkpoints. The checkpoints are represented by books, which the character has to collect. To emphasize the burlesque nature of our futuristic setting, we decided to realize the art style in a "cartoony" look with strong outlines. Deciding on a color scheme was representing an especially difficult task, since a library usually contains a high variation of books with different colored book covers. It was important to us to represent a huge library with its high variety of books, without dragging too much of the player's attention to the background.

THE DEVELOPMENT TEAM

Hien Quy Tran
General Concept
Level Design
GUI Implementation/Design
Hint Pop-Ups & Help Screens Implementation/Design
High Score Screen Implementation
Credit Screen Implementation/Design
Sound Implementation/Design

Jeremia Oelschläger
General Concept
Title Screen Design
Environment Design
Character Design/Animation
Help & Hint Screen Design
High Score Screen Design
Special FX Design

Josia Roncancio
General Concept
Core Mechanics Programming
Indicators Programming
Environment FX Implementation
Ghost Implementation
Achievements Implementation

Klemens Reckfort
General Concept
Core Mechanics Programming
Level Design
Character Animation
Special FX Implementation

Max Warsinke
General Concept
Core Mechanics Programming
Level Design
Camera Movement Programming
Parallax Implementation
Indicators Programming
Environment FX Implementation
Environment Design

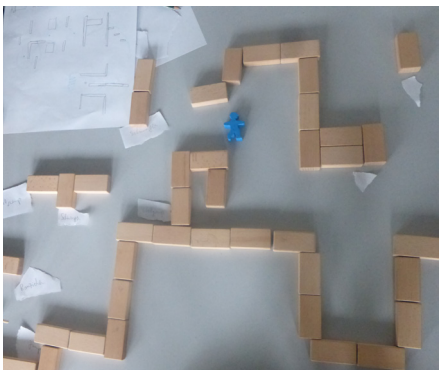
2. STARTING POINT

Before jumping into actual realization of the game's codes and assets, we were not only regularly meeting up for brainstorming sessions, but also for gaming sessions of our favorite 2D Jump & Run games. Two in particular were catching our attention: "Dustforce" from "Hitbox Team" and "Super Meat Boy" from "Team Meat". Both games are focusing on precise control and movement of the playable character. While "Dustforce" felt a bit too slow paced and "Super Meat Boy" a bit too fast paced, they still had a lot in common. We agreed, in terms of pace, that our game should feel like something in between of the two mentioned games. After several more sessions of brainstorming and exchange of ideas, a concept of "Armville", which we all agreed on, was born.

In-Game Screenshot of "Dustforce"

In-Game Screenshot of "Super Meat Boy"

Because this project represents our first team based game developing project, we did not want to assign individual tasks towards each other from the very beginning. We did not even precisely knew in which task area each one of us is good in or shared most interest in. Having said this, we were not expecting everyone to participate in every aspect of the game development, rather than expecting from each one of us to naturally participate in those fields, which he was interested in. This freedom and flexibility of participation and integration, helped us to have a wide inside look into game development and also encouraged us to involve each other in different task areas.



Brainstorming Session: Our first Pen & Paper Prototype

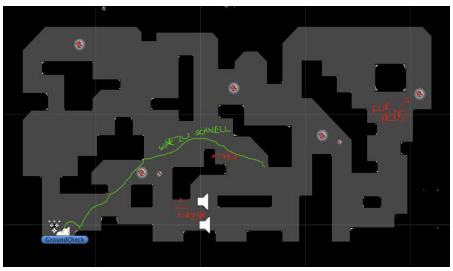
3. MY TASK AREA

The purpose of the following section is to give a quick overview of my task areas.

3.1 Level Design

The focus of our game lies in the movements of the character and the pacing of the game. With that in mind, we were giving not only the game mechanics, but also the level design a high importance from the very beginning in order to achieve the desired pacing and "feeling" of the game. It was important to us, to provide a level design, which rather feels well thought, than randomly generated. The planning of the level design gave us the opportunity to compose multiple routes with variations of linked movement to choreographies suitable for beginners and advanced players. In order to avoid a tile map editor look like, we decided to build the level before starting to create any graphical assets.

After we have commonly created a base for our level design, my individual tasks was to host and analyze plenty play testing sessions with several test subjects, which resulted into countless tweaks and optimization processes of several routes, as well as adding and removing alternate routes in order to reach an overall balanced level design to provide the best experience for beginners to highly advanced players.



Analysis of the Level Design

3.2 Character Design

The character design was challenged by the question of: "How to implement a huge arm to the character, without letting it look wrong?" Every one of us was crafting a mini maquette for inspirational purpose of the character design.

Mini Maquettes from Quy, Max and Jeremia. (from left to right)

3.3 GUI Implementation/Design

The graphical user interface is playing a major role in applying pressure to the player by providing instant information about the elapsed time and the game's progress. My primary task was to create a functional graphical user interface, which is clean and not distracting, while at the same time clear and self-explanatory. The layout and design had to fit and blend into the given art style.

In-Game Screenshot including final GUI and Help Screen

3.4 Hint Pop-Ups & Help Screens Implementation/Design

Due to the nature of our short termed, high score based time trail runner and the maximum scope of our project, we did not want to include an extensive tutorial, which would possibly take more time to finish in development than the demonstration level itself. Therefore, we decided to implement optional help screens, as well as non-frequent pop-ups of hints. The idea was to give the player enough time to explore the game mechanics herself, before providing any optional help. This allows the game to support players with hints and help screens only when really needed and desired. My task was the implementation of anything GUI related, and therefore also the implementation of the hint pop-ups and help screens. The major task I was facing, along the implementation, was the optimization in timing and structure. It was important to me, that the provided hints and help screens were clearly structured and not unnecessarily, excessively overloaded.

3.5 High Score Screen Implementation

The high score screen represents an indispensable part of any competitive, high score based video game. My task was not only the pure implementation of a high score screen, but also to find a way to make all operations within the high score screen, e.g. typing gamertags, etc., easy to control with the "Xbox 360" controller. Due to the high importance of the high score screen I was eager to implement and design a functional but also highly stylish high score screen.

In-Game Screenshot of the final High Score Screen

3.6 Credit Screen Implementation/Design

Each one of us was contributing important work and was essential in order to finish this project. It was important to me, that the credit screen would not set a fixed hierarchy in listing our names. Also, I wanted the credit screen to be discreet but stylish at the same time.

In-Game Screenshot of the final Credits Screen

3.7 Sound Implementation/Design

The major feature, which distinguishes games from other media, is the constant feedback the player can get. Gaming in particular is all about feedback, which means a decision or an input results in an output as the feedback. The feedback can be designed in many ways and can be communicated to the player in many ways. One of the most efficient ways, to return feedback is the feedback through sound. It is not only able to support and underline visual feedback, but is also able to communicate feedback in a way, that visual feedback is not always capable of.

As a musician and sound enthusiast I pay a lot of attention to the sound design and the feedback through sound in games. My task within this project was the implementation of the sound feedback and design of the sound assets. One of the major problems to solve was to find the right environment sound to emphasize the atmosphere of the quiet library, which stands in strong contrast to our action orientated fast paced side-scroller. The sound did not only have to be designed out of a functional perspective, but also had to fit the overall style of the game.

4. MY PERSONAL GOALS AND MILESTONES

Many professions within the game development were introduced to me for first time within this project. I have never developed a video game before and therefore did not know what the working pipeline could look like in game development. I did not have a sophisticated idea of what a game engine actually is, neither did I know what capabilities to expect from game engines in general. Also, I did not have much of programming knowledge at the beginning of this project.

Especially because I did not have much experience in game development and was lacking the general idea of how it looks like, my personal goal was to get a good overview of game development in general, including all processes of the common working pipeline. Although it might sound trivial to those who have got some experience working with game engines, I must admit, that I did not know what game engines actually are and wanted to gain a better understanding of them. This does not only include a better understanding of the general idea and concept, but also a better understanding of the possibilities they give us. Having a good idea of what a game engine is, was especially important to me, because it was representing the mandatory basic knowledge for me, which is necessary in order to be able to aim for further goals. In addition to that, I was new to programming and wanted to gain a better understanding of how programming within a game engine works. Having said this, I found it especially difficult to set myself any static goals in beforehand, because I was unsure, what specific goals I should be aiming for. I therefore did not exactly specify my personal goal, rather than defined it more general and left it being stretchable.

In other words: I wanted to find out my capabilities, rather than reaching a certain goal. In order to do so, I tried to assign myself to a broad variety of different task within every layer of the game development, in which I again tried to achieve the best possible result, instead of setting me a static goal. This of course required a lot of communication within the team, which was also new to me. This project was representing my first larger team project and I was curious about developing a game within a team.

Because I was always trying to raise the bar of my assigned tasks over time, I was able to see and experience the progress I was doing within this project. I was also often going through old codes and tasks as soon I had the feeling, that I could improve them in one or another way. This way I was not only able to take more advantage from recently gained knowledge, but could also practice them instantly.

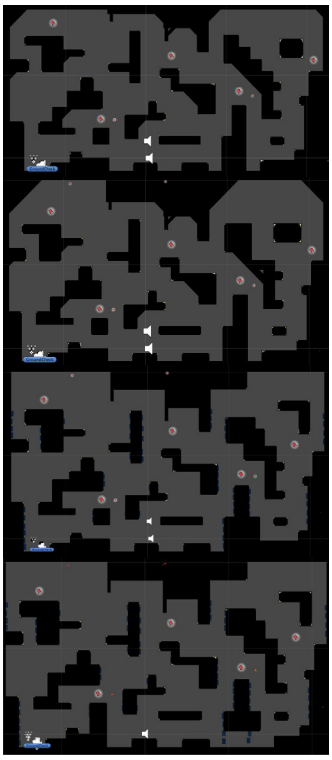
5. MY WORKFLOW

The following section will explain my task areas in more detail and give an approximate idea of how many hours I have been spending in those individual tasks by providing the hours in brackets at the beginning of every section.

5.1 Level Design (20h)

Intention:
We were aiming for a demonstration level, which would provide the best experience for the widest possible range of diverse player types. We did not want it to be too difficult for beginners and at the same time to be offering enough reasonable challenges for medium to advanced players. In order to maintain a high replay value, it was important to us to design a nonlinear demonstration level, which would offer various paths with each of them offering even more options on how to move through them. One of our other major tasks was to assure, that the level design would not provide any obvious superior path, which most of the players would end up going for automatically, rather than having the urge to replay the level in order to figure out their individual optimized path. Due to the limited scope of our project we were trying to use every space as efficient as possible.

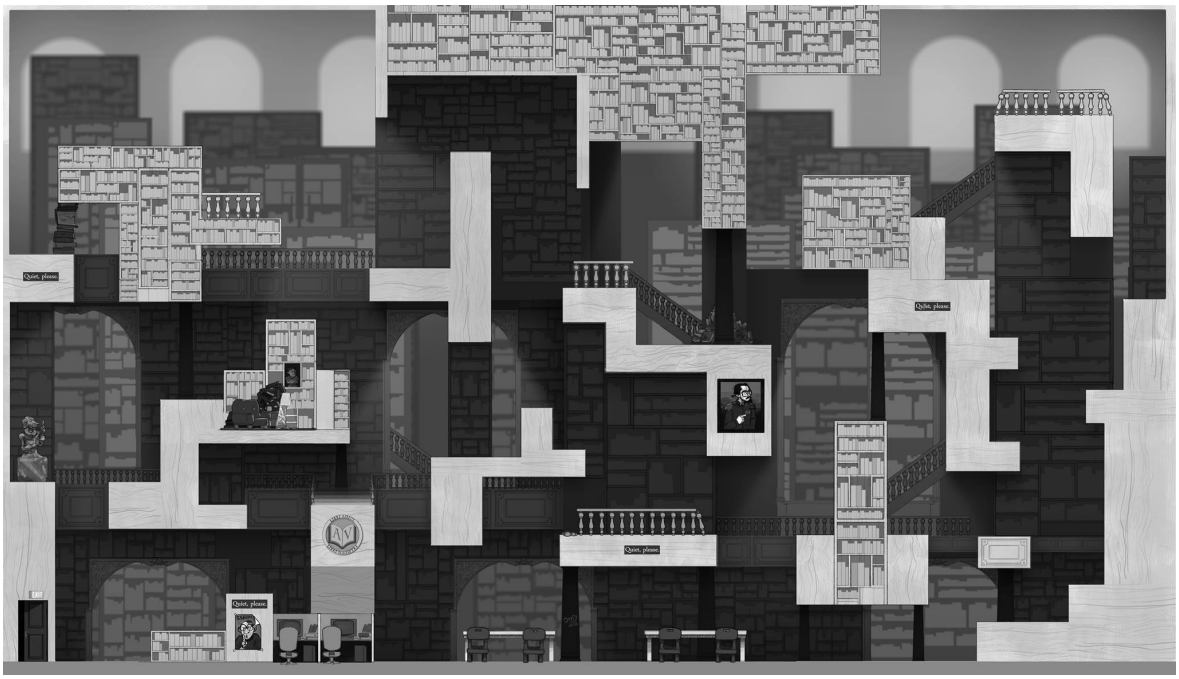
Working Process:
In order to gain a wide range of ideas and concepts for the level design, each one of us (Max Warsinke, Klemens Reckfort, Hien Quy Tran) was independently designing a personal prototype level without exchanging any ideas beforehand or affecting each other in any other way. That is why, we ended up with three highly diverse prototype levels, with each of them having major strengths and weaknesses. Our initial plan was to pick the one, which we would agree on to be most suitable for our project and use it as a common base for our level design in order to continue working on it. After comparing and analyzing each one of ours, we came to the conclusion, that we cannot decide on one particular prototype, because all of them had different major advantages but also disadvantages. With this new pool of shared knowledge and ideas, we tried to commonly create a level which would ultimately combines all the strengths of all three into one single design. Having set a level design we were all satisfied with, took us to the testing and improving stage.


Optimization Process of the Level Design

For the improving stage I have been hosting several play testing sessions, in which external subjects were playing our level for the first time while I was studying the whole session. I ended up in doing countless tweaks and optimization processes of the level design, as well as adding and removing whole areas in order to achieve an overall balanced level design. Every tweak and optimization process had always to be tested again by new external subjects, which in return was resulting in even more improvements in order to provide the best experience for beginners to advanced players. The several testing and improving units took me in total approximately about 20 hours.

Result:

Due to the intensive play testing and countless improvements, we achieved a fairly balanced level design. While the final demonstration level is perfectly suitable for medium and advanced players, it sometimes appears to be a bit too difficult for players, who are not familiar with game pads. We were aware of that issue, but we decided not to compromise by lowering the difficulty through level design, because it would have contradicted the overall idea of the game. We wanted the player to improve while playing our game with certain areas being set up, to push the necessity of improvement. A larger tutorial area would have helped some players to be able to jump right into the game with less frustration, but due to the limited scope of this project, we decided not to include it yet. With possible addition of more levels in the future, we might will include a small tutorial level, which could teach the controls and the game mechanics to the player step by step. Nevertheless, I am very satisfied with our demonstrational level design. It represents the design we were aiming for in the big picture. There is not a superior path, which makes all other paths redundant and I am confident, that we achieved a level design, which is delivering a highly diverse experience in terms of movement and problem solving all together.



Final Level Design (Art Design by Jeremia, Values by Max Warsinke)

Learning Effect:

Studying and going through countless optimization cycles, have demonstrated me, how little changes within the game design can have a huge impact on the game and change the way it is being played. It also underlined the importance of play testing in general. Fixing one design issue can cause another new unpredictable design issues to appear, which often can only be discovered by more intensive play testing. Also, it is important to let as many people as possible test the game, because the differences in the way they play games can result in huge differences in the player experience.

5.2 Character Design (5h)

Intention:
Our goal was to create a funny but also charming character which would emphasize the burlesque nature of our futuristic setting with his "cartoony" look. The biggest difficulty was to give our character an arm, which would be almost bigger than himself, in a believable and good looking way.


My unfinished Mini Maquette

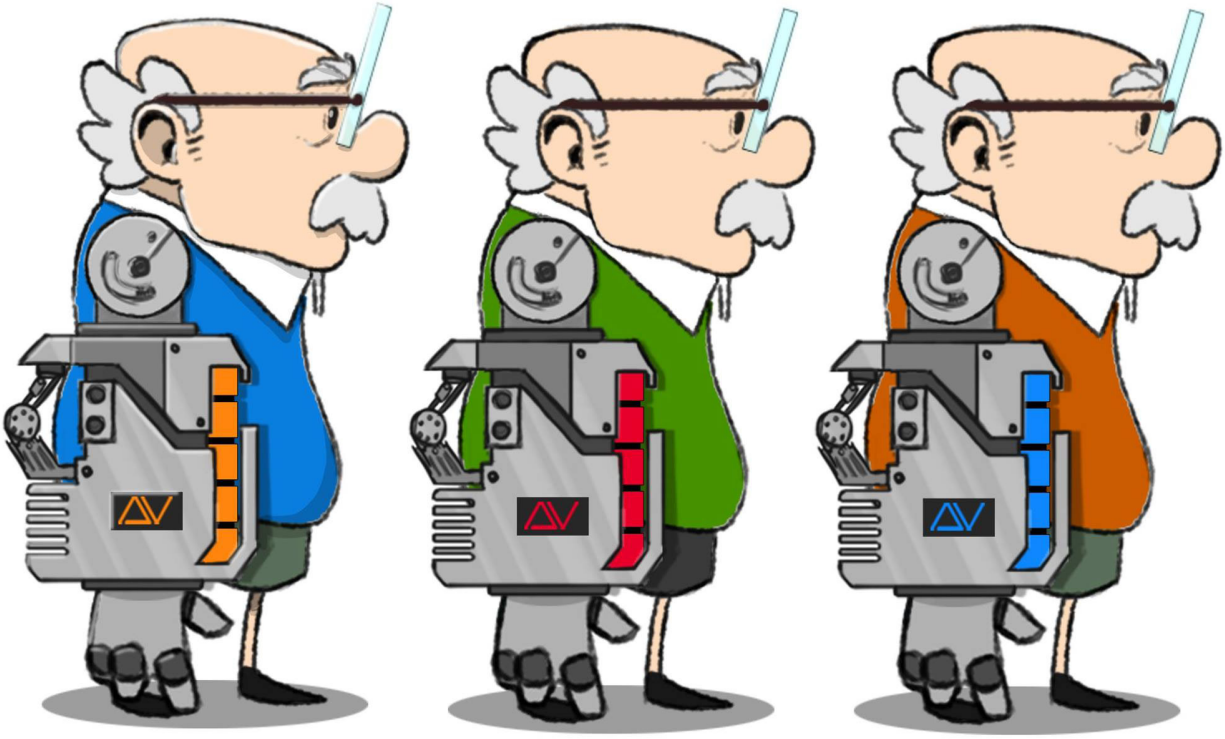
Working Process:
To gain a wide pool of ideas for inspiration, each one of us was creating a character mini maquette. After having analyzed our mini maquettes, we chose Jeremia's character, which represented an old librarian, as the most suitable. Each one of us was then drawing his own version of the librarian, based on Jeremia's character. We quickly ended up on agreeing that Jeremia's version of the librarian, named Archie, is the one we are going for.


Character Sillhouette by Jeremia

Result:
We are all very satisfied with the finalized version of Archie and could not think of any other better than him. He is charming and represents exactly the character we were aiming for.


Character Scribble by Jeremia

Learning Effect:
At first I was a bit skeptical, because I found it hard to imagine that an old, "fatty" man could perform insane jumps and acrobatic stunts. I was very surprised seeing it working so well. I have learned a valuable lesson, which I believed to know already, but apparently was not trusting enough: "It is often not about how realistic something is, but about how believable it is realized".


Character Design by Jeremia

Timer with Placeholder Graphics

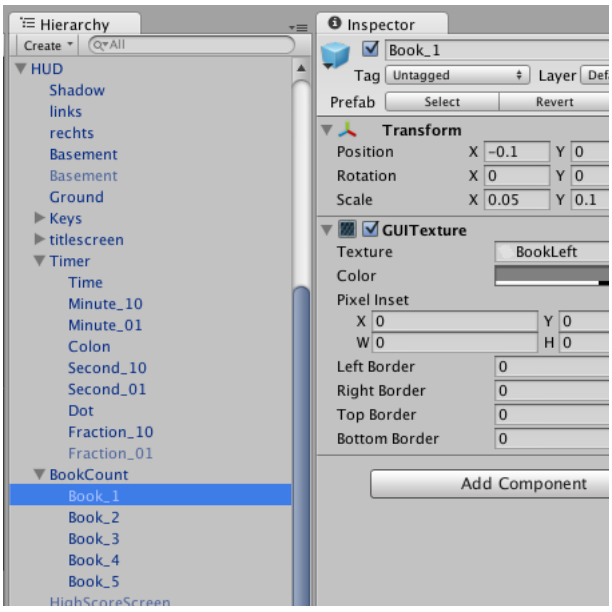
5.3 GUI Implementation/Design (15h)

Intention:
Because I did not have much of programming knowledge, I chose to start off with the implementation of something presumably easy. My first task within the implementation and programming was to create a functional graphical user interface. I wanted it to be clean, not distracting, while at the same time clear and self-explanatory. The layout and design had to fit and blend in to the art style.

Working Process:
At first I took the graphical asset of the collectable book, which was made by Jeremia, and created an icon out of it representing a "collected book" and an icon representing a "missing book". I then created a game object with five children, while adding a GUITexture to each one of them. After assigning the "missing book icon" to each of them, I scaled and positioned them properly to the upper right corner. Having done this, I then added a Script called "Timer.cs" to the parent, in which I was setting the conditions for a texture swap from the "missing book icon" to the "collected book icon" accordingly to the number of books collected. To make the graphical book count working multiple times, I just needed to add a condition, which would set the "book counter" back to zero and all GUITexture textures back to the 'missing book icon' texture on game reset.

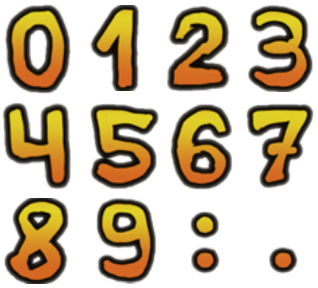
Graphical Assets based on Jeremia's Art

I added a float to the "Timer.cs" script, which is adding "deltaTime" to itself by each frame. This resulted in a float counting the seconds. In order to get this float into the time format I needed, I wrote three functions making this float resulting in three floats, one for minutes, one for seconds and one for fractions. Then I created eight children, one for each digit (including the tenners, singles and colons). Once again, I added a GUITexture to each one of the children and dragged then a placeholder graphic for the number zero into each of them. Having done this, I was able to properly scale and positioning them to the upper middle edge. I then created a public array list in the "Timer.cs" script named Numbers[] and dragged ten placeholder textures for the digits from '0' to '9' into this list. With those textures stored in the list, I was able to write a method into the "Timer.cs" script, which I called "FormatTime()". This method swapped every single texture accordingly to the digit of the formatted time. I then added a condition, which resets the textures and the timer to "00:00:00" on session restart.



Screenshot of Hierachy in Unity with
Game Objects for Timer and Book Count containing GUITextures

When the art design of our game became more solid, I started to design the graphical assets for the timer accordingly to the art style and replaced the placeholder graphics. At the end I deactivated the last fraction of the timer, because I did not see any use of it, since it was switching through all digits at a speed the human eye was not able to catch. The whole process from implementing and designing the timer and the book count took me approximately about 15 hours.



Graphical Assets for the Timer

Result:
The design of the GUI is very minimalistic and discreet, but still provides important information about the time and the players progress effectively. I am satisfied with the GUI implementation and design. It is not a major breakthrough, but its result is exactly what I was aiming for: "functional and neat".

Timer and Book Counter with Final Graphics

Learning Effect:
This was one of my first proper programming exercises and had the exactly right difficult to start on. In order to implement the timer I was using several array lists and loops, which I usually found difficult to use. This exercise was helping me in so many ways to set the ground base for programming and raised my interests for programming by more than anything else before.

5.4 Hint Pop-Ups & Help Screens Implementation/Design (25h)

Intention:
I wanted hints to pop up only when necessary. For that reason it was all about figuring out the right timing of when and what should pop up. Because not every player likes to be confronted with hint and help screens every time, I decided to implement an optional help screen, which would only pop up when the player decides to open it by pressing a button. The major task I was facing was the optimization in timing and structure. It was important to me, that the tutorials and hints provided were clear structured and not unnecessarily excessively overloaded.

Solution:
First, I created the graphical assets for the hint system, which I then set up as GUI Textures in several children. After that, I created a script named 'HUD.cs' which is dedicated to all the hint & help functions, including the timing of blending in and out the graphical assets. The timing pattern orientated itself on two different timers: One which is constantly monitoring how long the player has already been playing and another which is taking the time of how long ago the player has been stopped playing.

Graphical Assets for the Hint and Help System with my self-created Font

If the program does not register any input for a certain period of time, it would then assume, that the player is not playing anymore and had let go of the controller. The application would then suggest the next player to perform a restart instead continuing playing from the previous player's current game state in order to receive the full experience of the game. If the player does pick up the game without resetting the game, the application will assume that there is a chance, that it might be the same player continuing his current game and the suggestion for a restart would disappear after a short period. However, if the game does not get picked back up for an even longer period, the application will assume, that the chances are very low for the player to continue playing his game at a later stage and performs an automated reset back to the title screen.

On the other hand, if a player has been playing for a longer period of time without any progress, the application will assume, that the player might need some help, in which case a notification will pop up, suggesting the player the open up the help screen. The player is then able to open up the help screen by pressing the help button. Having the help screen open, he is able to scroll through the different sections and close it again. The help screen is placed to the upper left corner, so that the player is able to continue playing with the tutorials being displayed. Whenever the player manages to cover the playable character by the tutorial window, the application will lower the alpha channel of the tutorial window in order to make the character visible. If the player decides not to open the tutorial screen at all and is able to make further progress by collecting a book, the application will assume, that the player wants to figure out the game mechanics by herself and stops suggesting to open up the tutorial screen for a longer period of time.

In-Game Screenshot of open Help Screen

transparent Help Screen clearing the view, if Character is behind

Controller Mapping displayed, if Player doesn't know all functions

If a new player starts playing and does not move the right stick in order to punch or does not pull the triggers in order to grab, the application will assume, that the player does not know the controls. It will then show up the controller mapping in the upper middle section of the screen.

All those described responses of the application are mainly realized by finding the right child and set them active or inactivate in a certain time pattern. In order to blend the hints in and out smoothly, I was decreasing and increasing the alpha channel of the textures.

Until I got everything timed like I wanted it to be, I had to observe a lot of players playing the game for the first time, being confronted with the new game mechanics. I had to tweak the timing over and over again until the timing structure grew to something more and more complex. Just to give an example: I had to differentiate the necessary idle time until reset to title screen, while having the tutorial windows open, from having them closed, because some people would take the time reading all pages without moving the character around.

Result:
The pop-up hints and especially the help screens are compensating the lack of a tutorial level enormously. I managed to integrate and time the support in a way, so that it would only show up when necessary. And when it does, it is clearly structured and does not overflood the player with unnecessary information. I am very satisfied with the result and got a lot of positive feedback by the testers.

Learning Effect:
After I have finished implementing the graphical user interface, I felt like implementing a "smart" help system would be the ideal next step in challenge and difficulty. It did not only help me by repeating a lot of the techniques I have been using for the timer implementation, but also confronted me with new techniques to learn in order to solve the problems. The more complex interconnection of all variables forced me to be more careful in planing and required good overview over all parts of the code.

5.5 High Score Screen Implementation (35h)

Intention:
My intention was not only to implement a good looking high score screen, but also to find a way around to make all operations, e.g. typing gamertags, easy to perform with the Xbox 360 Controllers. It was my task to implement and design a functional but also highly stylish high score screen, which saves and loads up scores from previous sessions automatically. I also wanted to implement a private high score list, which could be activated by pressing a secret button combination, in case want to play during a fair, without spamming the public high score list.

Solution:
Before the game offers the player to type in a gamer tag after she completed a run, it has to compare the time with the times of the high score list. For this task it is comparing the unformatted floats, which contains the time value formatted to seconds. It starts comparing the scores starting with the top score all the way down to the last score, by the us of a loop. As soon it realizes that a recently set time is smaller than one of the times in the high score list, it stops the current operation and jumps to the next loop. Now all the times of the high score list, which are lower than the current time, have to move one position down, in order to make space for the current time. Once the new high score is set in place, all the listed times just need to be changed in to the right format and displayed through SpriteRenderers, which are realized in a similar way as the formation of the GUI timer.

High Score Screen with Placeholder Graphics

After setting a new high score, the application will wait for the player to enter her gamer tag. Because I do think it would be annoying and taking much time to scroll through all letters of the alphabet, I wanted to work around it. Since the playable character is able to punch any obstacles, I decided to design punchable letters, realized by twenty-three colliders on the ground with each of them representing a letter of the alphabet. As soon Archie's hand is colliding with one of them, the corresponding letter will appear.

I have been then adding an achievement high score list, which will only appear, when at least one of the achievement conditions are fulfilled (For example: Finishing the game without using grab.), as well as a private high score list, which can be activated by pressing a secret code. (LS + RS -> back button) In order to make the transfer work I had to implement some methods to swap the high score lists. In all cases it works by the same idea: Before the formatting and ranking happens, the public highs score list is saved into a temporary list, and the desired high score list is being loaded from another temporary list. After formatting and saving, the desired list is loaded back to the temporary list and the public high score list gets loaded back to the actual high score list.

Graphical Assets

Additional to that, I have implemented an automated save and load function. It is realized through the Unity playerPrefs, which are loaded at application start up and being saved after every new high score entry. This makes sure, that the high scores are staying after rebooting the game.

Later, when we had to reset the high score list several times. I figured out, that it would be too complicated and can take a lot of time to find the file containing the playerPrefs in order to delete it. (It is stored in the Windows registry or in the OSX system files.) For that reason I have implemented a secret button combination, which deletes the active high score list. (LB, Y, RB, A, LB, Y, RB, A)

Result:
The high score screen is not only very functional and stylish at the same time, but also includes gimmicks like the secret high score lists. Operating within the high score screen has been optimized and includes an auto save function, making it really comfortable to use. I am fully satisfied with the result.

Highscore Screen with Graphic Assets from Jeremia

Learning Effect:
Programming the high score screen was a bigger task for me to take on. It got very fast very confusing with all the loops and all the conditions. Sometimes I had difficulties to make the loops working properly. Implementing the alternative high score lists was challenging me. I am very happy, that it all worked out well at the end.

5.6 Credit Screen Implementation/Design (5h)

Intention:
Each one of us has contributed important work and was essential in every way to finish this project. It was important to me, to create a credit screen without setting any hierarchy for listing our names. Also, I wanted the credit screen to be discreet but stylish at the same time.

Solution:
The high score screen was providing the optimum space for listing our names. Instead of designing yet another screen just for the credits, the better solution was to remove the high scores from the already great high score screen and replace them with our names. That way I was able to keep the credit screen consistent to our art style.

I was designing the names in Photoshop and pixelated them to multiple degrees. I was then able, to use those frames to create a flicker animation for each name in Unity. After that, I added a script, which removes the high scores after a certain period of time and replaces them with those animations. The timing depends on many factors, like for example whether the character is moving or not and whether the player has set a record or not. Once this was working, I added a random variable to the script in order to make the names appear in a random order. It works as following: Starting with the first name, each name, gets a random number from one to five. This number is representing the position in the credit list. If the position has been already taken by another name, it will get a new random number. This happens, until every name has got a random number, which is not taken by another name. The names are then placed in the credit list accordingly. Each time the credit screen appears, the positions are shuffled new.

Animaton: from pixelated names to sharp names

Result:
The credit screen is working as supposed to and is discreet but stylish at the same time.

High Scores replaced by Credits (including Graphic Assets from Jeremia)

Learning Effect:
The credit screen was the last thing I was implementing and did not represent a too difficult task to realize. Because I was using the animation system of unity for the first time, I was surprised how easy it was to use. The easy implementation of the credit screen was demonstrating me my work progress in terms of gained skills and learning effect over the whole project. This made me very happy.

5.7 Sound Implementation/Design (25h)

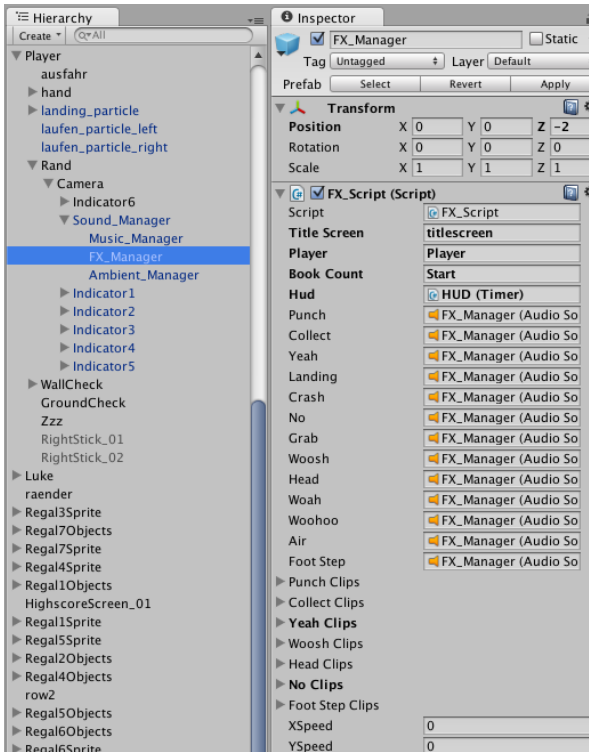
Intention:
As a musician and sound enthusiast I pay a lot of attention to the sound design and the feedback sounds of games. One of my tasks was the implementation and design of our game sound. It was not easy to think of an environment sound, which would reflect the atmosphere of a quiet library, but also emphasizes the action orientated fast paced game play of our side-scroller. The sound did not only have to be designed out of a functional perspective, but also had to fit the overall style of the game.

The possibilities of video games as an interactive medium were always fascinating me. Unlike movies, everything can be influenced and altered by the player. Instead of just being purely a consumer, the player can become producer to a certain degree. I like the idea of giving feedback to the player in every aspect of a game. Not only visuals and special sound effects, but almost everything should be a reaction of the player's action. This is, what I believe, the major strength of video games and what distinguishes it most from other form of media or non-interactive art.

Solution:

Because the art style was not definite at the beginning, I was not able to design the finalized sound assets for the game. Instead, I had to start implementing sounds by using placeholder sounds. The techniques I was using to implement sound and tell the application when to trigger which, can be grouped into three categories.

For the first group of sounds, I needed detailed information about every collision the playable character is confronted with. In order to get this required information I had to add a script with a "void OnCollisionEnter2D()" method to the playable character. This ensures, that all the information about every collision can be handed over to my sound script. Due to this method, my sound script does not only know, when the playable character is colliding, but also with what velocity it is colliding. By knowing the relative velocity, my sound script is able to distinguish between different cases of the collision's direction and strength. The collision's direction is important to be able to differentiate between cases like for example: hitting the head, landing on the feet or running into a wall. The collision's strength is easily linked to the volume of the triggered sound, so that a greater impact is louder than a smaller impact.



Hierarchy and Inspector of my "Sound_Manager" in Unity

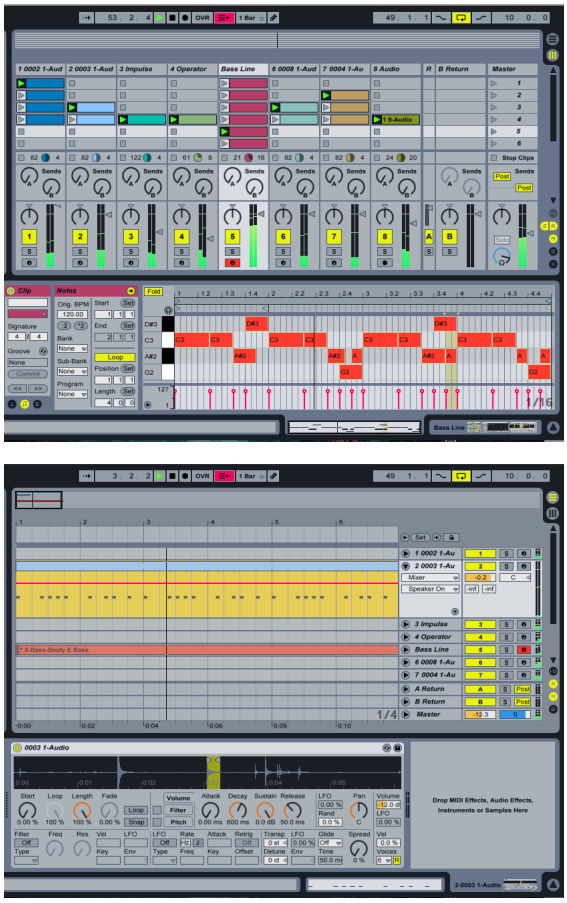
The whip sound of the playable character's arm and the air stream sound of the playable character's body while in air are forming the second group of sounds. For those sounds the opposite applies: not the collision's velocity matters, but the constant velocity, while being in a state of not colliding, matters. The constant velocity of the playable character and his arm can be read out of the "Rigidbody2D.velocity" variable. If the velocity is high enough, a sound will be triggered and the volume, as well as the pitch is adjusted accordingly.

The last and probably the most simple group of sounds are the sounds for reaching checkpoints. The sounds are simply triggered, when the playable character collides with a book or with the exit.

Having implemented all sound effects, I decided to implement a sound track, which would be dynamic and adapting to the pace and progress of every individual player. I was implementing a dynamic music system with placeholder sounds, which I later replaced by the final music assets.

The music is reflecting the progress of the game by increasing the intensity of the bass line with every collected book. In order to apply even more pressure, drums and percussion is added at a later stage. Because the playable character is usually faster in air, I wanted to emphasize it by letting the percussion only play while in air. The percussion is added after collecting books, but also fades out after a while again, if the player stops collecting books.

When the art style was finally set, I composed a sound track accordingly. I then split the sound track into several audio lines. I ended up with six bass lines, one drum line and one percussion line, each of them running on hundred-forty beats per minutes. The sound track starts off with a solo bass line to reflect the atmosphere of a quiet library and ultimately ends up with a composition including drums and percussion.



Ableton Live's Session (top) and Arrangement View (bottom)

I then browsed through our preselected sound library and picked the suitable ones, which I edited in order to fit the art style. Mostly, I had to reduce on the high pitches in order to make the assets sound more dull. This is because, the library is mostly dominated by big wooden objects and heavy books, which would swallow some of the high pitch frequency. I also had to record some missing sounds and dub our playable character with my own voice. Having adjusted the sound effects consistently, I was able to replace the placeholder sounds for good.

At last, I was designing additional sounds for the Archie ghost and the high score screen. I was modulating sounds by using the Operator in Ableton Live and added some sound effects afterwards. The ghost sound was then attached to the ghost itself as a 3D sound. I then adjusted the 3D sound settings of the ghost in Unity, in order to get the best balanced sound effect when the ghost floats over the screen. I then attached another modulated sound as a 3D sound to the high score screen and adjusted the settings accordingly. Since the high score screen sound is set as a 3D sound, the player will only hear it, when having reached it.

Result:

I am very happy about the result and I am sure, that I succeeded in catching the right atmosphere and matching the style of our game. I know, that there is a lot of room for further improvements, which I couldn't tackle on due to the lack of time. The sound is working well enough and there were more urgent tasks to finish at the end of the project.

Learning Effect:

As a musician, this was a particular interesting task for me. Because I enjoy composing music and designing sounds, I was always very curious about the design and implementation of sound in games. After this project, I have learned a lot about the possibilities of sound design in games. This experience was very inspiring and the possibilities are overwhelming. Designing paired with the implementation of sound is adding a complete new dimension for creativity and problem solving. This has only increased my interest for sound design. This task was a lot of fun and I cannot wait to continue experimenting in this area.

6. RECAP

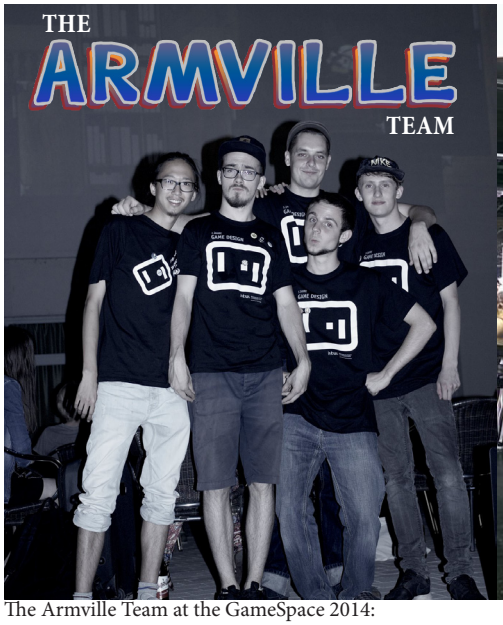
At the beginning of this project I did not know what to expect at all. I had not much of an idea how to approach this project. I was also lacking strong programming skills and have never worked with any game engine before. Now, having finished this project, I can say, that this project was very helpful for me. Not only that I am very satisfied about the result of our game, but more so about the progress and experience I have been gaining with this project. I was able to gain more knowledge about game development than I initially thought I would. Especially is to mention my great leap forward in programming and understanding of game engines. Because I have assigned myself into a broad variety of tasks, I was able to widen my knowledge especially horizontally. I have a much more defined idea of the working pipeline in game development. This helped me to figure out, which parts of development I do enjoy most, and which I would rather like to avoid. Knowing the own interests represents a great advantage for further projects, education and specializing. This enables me in future projects to build up my knowledge vertically and go into more depth.

One of the greatest discoveries of this project, was the fun I had with programming. I never shared a lot of interest for programming, but during this project I realized, that this was the part of the project I was actually enjoying most. I would go that far to say, that I have learned enough to be able to program my own small "games" now. Especially, when it came to the implementation of sound via code, I was overwhelmed by the creative possibilities of programming. This was really removing the scales from my eyes. Being able to design my own sounds for implementation was just the icing on the cake and I am now even more curious about the possibilities in sound design than ever before.

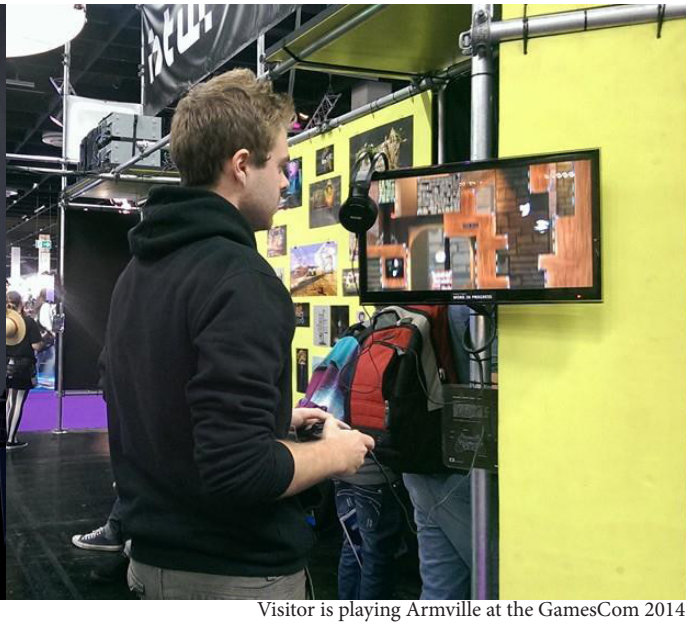
Looking back at my expectations, personal goals and milestones, I would mark this project as a great success. Although I was not able to set a specific goal and milestone from the very beginning, I feel like having surpassed my expectations by far.

I have a much clearer image of what to expect for the next project now. This allows me to have a better orientation on how to prepare for future projects and enables me to aim for further goals. I would like to focus on programming and sound design for the next project, while hoping, that I will be able to tackle bigger programming problems and modulate my sounds in a more defined way.

Happy End! :)



The Armville Team at the GameSpace 2014:
Quy, Jeremia, Klemens, Josia, Max (from left to right)



Visitor is playing Armville at the GamesCom 2014