Designing HTML5 2D platformer for web cam based motion games

by

Tim Ziegenbein

LIU-IDA/LITH-EX-A--15/051--SE

2015-10-09
Final Thesis

Designing HTML5 2D platformer for web cam based motion games

by

Tim Ziegenbein

LIU-IDA/ LITH-EX-A--15/051--SE

2015-10-09

Supervisor: Aseel Berglund, IDA, Linköping University
Examiner: Johan Åberg, IDA, Linköping University
Movement interaction in games has over the last years been a main subject in the area of game design and recent motion based interaction systems provide many improvements over earlier systems. Research has proven that motion based games provide several benefits over general games, they increase the players’ engagement level and may induce both mental and physical health benefits. Still, not much research has been conducted in the area of motion based platform games, even though the platform game genre is one of the most popular game genres. In this thesis, a case study is conducted in order to extract design principles and guidelines applicable for motion based platform games. The study includes implementation of a browser-based 2D platform game that uses the web camera to provide motion based interaction. One of the conclusions in this study indicates that many game features from ordinary platform games only require minor modifications to be applicable for motion based platform games.
I would like to thank my supervisors Erik and Aseel for providing great suggestions and constructive criticism to help me throughout the process of this study. I really appreciate your involvement in the project and I had a lot of fun testing out new game features with you.

I would also like to thank my family and my girlfriend for encouraging and supporting me through the time of this project as well as my whole education.
## Contents

Abstract  

Acknowledgements  

Contents  

List of Figures  

List of Tables  

1 Introduction  
  1.1 Motivation  
  1.2 Aim  
  1.3 Research Question  
  1.4 Delimitation  
  1.5 Background  

2 Related Work  
  2.1 Video Games and Entertainment  
  2.2 Platform Games  
    2.2.1 Game Play and Game Mechanics  
    2.2.1.1 Platform Game Components and Level Structure  
    2.2.1.2 Player Experience and Fun in 2D Platform Games  
    2.2.1.3 Procedural Level Generation  
    2.2.1.4 Challenge and Rewards  
    2.2.1.5 Player Character Movement  
    2.2.1.6 Common Tasks and Obstacles  
    2.2.2 Game Interface  
    2.2.3 Graphics  
    2.2.4 Audio Elements  
  2.3 Motion Based Games  
    2.3.1 Motion Based Interactions  
    2.3.2 Game Play & Game Mechanics  

3 Method  
  3.1 The Planning Phase  
  3.2 The Iterative Phase
## Contents

4 Results

4.1 The Planning Phase ........................................... 32
4.2 The Iterative Phase ........................................... 33
  4.2.1 Iteration 1 ........................................... 33
  4.2.1.1 Tasks ........................................... 33
  4.2.1.2 Evaluation ........................................... 38
  4.2.2 Iteration 2 ........................................... 41
  4.2.2.1 Tasks ........................................... 42
  4.2.2.2 Evaluation ........................................... 47
  4.2.3 Iteration 3 ........................................... 49
  4.2.3.1 Tasks ........................................... 49
  4.2.3.2 Evaluation ........................................... 59

5 Discussion ....................................................... 61
  5.1 Results ....................................................... 61
  5.2 Method ....................................................... 65
  5.3 The Work in a Wider Context ................................. 66

6 Conclusion ....................................................... 67
List of Figures

1.1 Initial Game ................................................. 3
2.1 Framework Model ............................................ 12
2.2 Procedurally Generated Platformer ......................... 14
2.3 Pendulum Obstacle ........................................ 18
2.4 Unstable Floor Obstacle ................................... 18
3.1 UXDesign .................................................... 29
4.1 Level Complete State ....................................... 34
4.2 Flow Chart .................................................. 36
4.3 Character and Environment Size Comparison ................ 36
4.4 Tiled Editor Screen Capture ................................ 38
4.5 Motion Zone Layout - Iteration 1 ......................... 39
4.6 Example of Upward and Downward Paths ................. 42
4.7 Level Block Structure ..................................... 43
4.8 Example of Top-right block and Bot-right block .......... 45
4.9 Motion Zone Layout - Iteration 2 ......................... 46
4.10 Game Progression Example ............................... 51
4.11 Motion Zone Layout - Iteration 3 ......................... 52
4.12 Spear Pendulum Example .................................. 53
4.13 In game tutorial - Run right sequence .................. 54
4.14 In game tutorial - Jump sequence ........................ 55
4.15 Bonus Level ................................................ 56
4.16 Pause Menu ................................................. 57
4.17 Image Sharing .............................................. 59
List of Tables

2.1 Game Heuristics Presented by Federoff ................................. 6
2.2 GameFlow Model .......................................................... 9
2.3 Statistically Significant Correlations ................................. 13
2.4 Movement Based Interaction Guidelines - Cluster 1 ............ 24
2.5 Movement Based Interaction Guidelines - Cluster 2 ............ 26
2.6 Movement Based Interaction Guidelines - Cluster 3 ............ 27

4.1 Case Study Plan ............................................................. 32
4.2 Tasks - Iteration 1 ......................................................... 39
4.3 Heuristics and Guidelines evaluation - Iteration 1 ............... 40
4.4 Tasks - Iteration 2 ......................................................... 47
4.5 Heuristics and Guidelines evaluation - Iteration 2 ............... 47
4.6 Tasks - Iteration 3 ......................................................... 58
4.7 Heuristics and Guidelines evaluation - Iteration 3 ............... 59
Chapter 1

Introduction

1.1 Motivation

In the early era of video games, the platform game genre had one of the greatest market shares ever. Since then, the market share for platform game have dropped considerably but the genre had a huge upswing in popularity since the incorporation of modern mobile phone games. This indicates that combining platform games with the new concept of motion based interaction could further renew the popularity of the genre. In this thesis, I am hoping to provide design principles and guidelines for motion based platform games that can be applied to enhance the entertainment level in such games. Consequently a proof of concept browser-based platform game will be created with a technique that allow the web camera to be used for interaction. Furthermore, a successful motion based platform game could have several benefits over general platform games. Research has proven that motion based games may affect players in a way that increases both physical and mental health. Motion based games is also known to increase the engagement level of the players compared to non-movement based games. Thus, combining one of the most popular game genres ever with motion based interaction could possibly result in highly entertaining game with several benefits over standard platform games.

1.2 Aim

The primary aim of this thesis is to establish how 2D platform games that relies on motion based interaction should be designed and to examine how components frequently used in regular 2D platform games copes with motion based 2D platform games. The project will include the development of a 2D platformer that uses motion based interaction.
Chapter 1. Introduction

1.3 Research Question

The main question at issue is:

- How to design action-based 2D platform games that relies on motion based interactions to ensure player satisfaction?

In this case, player satisfaction implies that the game should be enjoyable with minimal player frustration due to the limitations of the interaction scheme.

1.4 Delimitation

This thesis addresses the interplay between motion based interaction and 2D platform games. Even though the related work chapter partly covers game design in general, the evaluation and results of the project was mainly intended to be viable for the 2D platform game genre.

The motion based interaction in this study relies on an existing web camera plug-in. While results and conclusions may apply for other types of motion based interaction techniques, the research was based on the interaction technique described in the background section.

1.5 Background

The work in this study is based on an existing 2D platform game project which is meant to be extended in this project and later be part of a bigger game structure. Initially, the game consisted of one level that was composed by platforms, foreground objects and background objects. The player character was controlled by physical interaction using a web camera as detection device. The game also included one type of collectibles and one type of monsters. The monsters could not be interacted with and no level goal existed. Figure 1.1 displays a snapshot from the initial stage of the game, prior any modifications. The red contours is a shadow sprite, a real-time feedback displaying the position of the actual player. When applying constant motion in any of the four predefined motion zones indicated by the black lines, actions are performed by the character. In this case, when activating zone one or zone two, a jump action is performed by the avatar. Zone three will, when triggered, cause the avatar to move horizontally in the left direction while zone four will move the character to the right. The game was developed using the
Chapter 1. *Introduction*

desktop and mobile HTML5 game framework Phaser [1] and the levels were designed utilizing the Tiled [2] map editor.

![Figure 1.1: A snapshot from the original game. The fourth zone is highlighted, indicating that the zone has been activated and the avatar will consequently move to the right. The stars are collectibles representing score values.](image-url)
Chapter 2

Related Work

This section will cover scientific and industrial literature in the area of video games, platform games and motion based games.

2.1 Video Games and Entertainment

The focus of this thesis lies on platform games and motion based camera interaction. However, some concepts and standards regarding video games in general are relevant when analyzing design and entertainment value in platform games.

A well-established structuring in Human-computer Interaction of video games was presented by Clanton. The area was suggested to be divided into three levels; game interface, game mechanics and game play. Game interface includes the physical interaction, such as how to control a game, as well as software controls presented on the screen. Game mechanics relate to the physics of the game, e.g. in what manner a player is allowed to navigate through the game environment. Game play represents the game design regarding challenges in games, i.e. obstacles that a player has to overcome to achieve a goal. This structuring encapsulates the usability issues and all three levels strongly relates to player satisfaction of games. [3]

A set of heuristics and usability guidelines intended to be used in the development and evaluation of video games was introduced by Federoff. By reviewing literature and interviewing game developers in a leading game development company suitable heuristics could be elected and assembled in a table (see Table 2.1). The heuristics are divided into three different levels of HCI as introduced by Clanton. Appliance of Federoff’s heuristics as a usability evaluation tool in game development processes would facilitate the production of successful and enjoying games. Federoff also underlined the
importance of prototyping and testing in game development processes to minimize the risk of failure. [4]

<table>
<thead>
<tr>
<th>Category</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game Interface</td>
<td>Controls should be customizable and default to industry standard settings</td>
</tr>
<tr>
<td>Game Interface</td>
<td>Controls should be intuitive and mapped in a natural way</td>
</tr>
<tr>
<td>Game Interface</td>
<td>Minimize control options</td>
</tr>
<tr>
<td>Game Interface</td>
<td>The interface should be as non-intrusive as possible</td>
</tr>
<tr>
<td>Game Interface</td>
<td>For PC games, consider hiding the main computer interface during game play</td>
</tr>
<tr>
<td>Game Interface</td>
<td>A player should always be able to identify their score/status in the game</td>
</tr>
<tr>
<td>Game Interface</td>
<td>Follow the trends set by the gaming community to shorten the learning curve</td>
</tr>
<tr>
<td>Game Interface</td>
<td>Interfaces should be consistent in control, color, typography, and dialog design</td>
</tr>
<tr>
<td>Game Interface</td>
<td>Minimize the menu layers of an interface</td>
</tr>
<tr>
<td>Game Interface</td>
<td>Use sound to provide meaningful feedback</td>
</tr>
<tr>
<td>Game Interface</td>
<td>Do not expect the user to read a manual</td>
</tr>
<tr>
<td>Game Interface</td>
<td>Provide means for error prevention and recovery through the use of warning messages</td>
</tr>
<tr>
<td>Game Interface</td>
<td>Players should be able to save games in different states.</td>
</tr>
<tr>
<td>Game Interface and Play</td>
<td>Art should speak to its function</td>
</tr>
<tr>
<td>Game Mechanics</td>
<td>Mechanics should feel natural and have correct weight and momentum</td>
</tr>
<tr>
<td>Game Mechanics</td>
<td>Feedback should be given immediately to display user control</td>
</tr>
<tr>
<td>Game Mechanics and Play</td>
<td>Get the player involved quickly and easily</td>
</tr>
<tr>
<td>Game Play</td>
<td>There should be a clear overriding goal of the game presented early</td>
</tr>
<tr>
<td>Game Play</td>
<td>There should be variable difficulty level</td>
</tr>
<tr>
<td>Game Play</td>
<td>There should be multiple goals on each level</td>
</tr>
<tr>
<td>Game Play</td>
<td>“A good game should be easy to learn and hard to master” (Nolan Bushnell)</td>
</tr>
<tr>
<td>Game Play</td>
<td>The game should have an unexpected outcome</td>
</tr>
<tr>
<td>Game Play</td>
<td>Artificial intelligence should be reasonable yet unpredictable</td>
</tr>
<tr>
<td>Game Play</td>
<td>Game play should be balanced so that there is no definite way to win</td>
</tr>
<tr>
<td>Game Play</td>
<td>Play should be fair</td>
</tr>
<tr>
<td>Game Play</td>
<td>The game should give hints, but not too many</td>
</tr>
<tr>
<td>Game Play</td>
<td>The game should give rewards</td>
</tr>
<tr>
<td>Game Play</td>
<td>Pace the game to apply pressure to, but not frustrate the player</td>
</tr>
<tr>
<td>Game Play</td>
<td>Provide an interesting and absorbing tutorial</td>
</tr>
<tr>
<td>Game Play</td>
<td>Allow players to build content</td>
</tr>
<tr>
<td>Game Play</td>
<td>Make the game replayable</td>
</tr>
<tr>
<td>Game Play</td>
<td>Create a great storyline</td>
</tr>
</tbody>
</table>
Chapter 2. Related Work

<table>
<thead>
<tr>
<th>Game Play</th>
<th>There must not be any single optimal winning strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game Play</td>
<td>Should use visual and audio effects to arouse interest</td>
</tr>
<tr>
<td>Game Play</td>
<td>Include a lot of interactive props for the player to interact with</td>
</tr>
<tr>
<td>Game Play</td>
<td>Teach skills early that you expect the players to use later</td>
</tr>
<tr>
<td>Game Play</td>
<td>Design for multiple paths through the game</td>
</tr>
<tr>
<td>Game Play</td>
<td>One reward of playing should be the acquisition of skill</td>
</tr>
<tr>
<td>Game Play</td>
<td>Build as though the world is going on whether your character is there or not</td>
</tr>
<tr>
<td>Game Play</td>
<td>If the game cannot be modeless, it should feel modeless to the player</td>
</tr>
</tbody>
</table>

Table 2.1: Game heuristics presented by Federoff [4].

Sweetser & Wyeth have composed a model by combining various game heuristics to further ease the process of evaluating player enjoyment in video games, the GameFlow model [5]. GameFlow was developed by using the famous concept of Flow as a foundation [5]. The flow theory investigates and provides evidence to what makes experiences satisfying; an optimal satisfying experience is called ‘flow’. The concept was introduced by psychologist Csikszentmihalyi who conducted over 30 years of research in the area [6].

Game heuristics and usability guidelines were matched with core elements of flow to create the GameFlow model represented in Table 2.2. Sweetser & Wyeth used the model to evaluate two games in the real-time strategy game genre, one successful and one less successful game. The result proved that the successful game consistently fulfilled a larger amount of GameFlow criteria than the less successful game, thus validating the GameFlow model. Summarizing, challenge is one of the most important aspects of the game for keeping the player focused and engaged in the game. The challenge requires however to be carefully balanced to not exceed the skill level of the player. Tasks in the game should have clear goals and a greedy reward system is preferred. [5]
## Chapter 2. Related Work

<table>
<thead>
<tr>
<th><strong>Concentration</strong></th>
<th>Games should require concentration and the player should be able to concentrate on the game</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• games should provide a lot of stimuli from different sources</td>
</tr>
<tr>
<td></td>
<td>• games must provide stimuli that are worth attending to</td>
</tr>
<tr>
<td></td>
<td>• games should quickly grab the players’ attention and maintain their focus throughout the game</td>
</tr>
<tr>
<td></td>
<td>• players shouldn’t be burdened with tasks that don’t feel important</td>
</tr>
<tr>
<td></td>
<td>• games should have a high workload, while still being appropriate for the players’ perceptual, cognitive, and memory limits</td>
</tr>
<tr>
<td></td>
<td>• players should not be distracted from tasks that they want or need to concentrate on</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Challenge</strong></th>
<th>Games should be sufficiently challenging and match the player’s skill level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• challenges in games must match the players’ skill levels</td>
</tr>
<tr>
<td></td>
<td>• games should provide different levels of challenge for different players</td>
</tr>
<tr>
<td></td>
<td>• the level of challenge should increase as the player progresses through the game and increases their skill level</td>
</tr>
<tr>
<td></td>
<td>• games should provide new challenges at an appropriate pace</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Player Skills</strong></th>
<th>Games must support player skill development and mastery</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• players should be able to start playing the game without reading the manual</td>
</tr>
<tr>
<td></td>
<td>• learning the game should not be boring, but be part of the fun</td>
</tr>
<tr>
<td></td>
<td>• games should include online help so players don’t need to exit the game</td>
</tr>
<tr>
<td></td>
<td>• players should be taught to play the game through tutorials or initial levels that feel like playing the game</td>
</tr>
<tr>
<td></td>
<td>• games should increase the players’ skills at an appropriate pace as they progress through the game</td>
</tr>
<tr>
<td></td>
<td>• players should be rewarded appropriately for their effort and skill development</td>
</tr>
<tr>
<td></td>
<td>• game interfaces and mechanics should be easy to learn and use</td>
</tr>
<tr>
<td><strong>Control</strong></td>
<td>Players should feel a sense of control over their actions in the game</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>• players should feel a sense of control over their characters or units and their movements and interactions in the game world</td>
</tr>
<tr>
<td></td>
<td>• players should feel a sense of control over the game interface and input devices</td>
</tr>
<tr>
<td></td>
<td>• players should feel a sense of control over the game shell (starting, stopping, saving, etc.)</td>
</tr>
<tr>
<td></td>
<td>• players should not be able to make errors that are detrimental to the game and should be supported in recovering from errors</td>
</tr>
<tr>
<td></td>
<td>• players should feel a sense of control and impact onto the game world (like their actions matter and they are shaping the game world)</td>
</tr>
<tr>
<td></td>
<td>• players should feel a sense of control over the actions that they take and the strategies that they use and that they are free to play the game the way that they want (not simply discovering actions and strategies planned by the game developers)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Clear Goals</strong></th>
<th>Games should provide the player with clear goals at appropriate times</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• overriding goals should be clear and presented early</td>
</tr>
<tr>
<td></td>
<td>• intermediate goals should be clear and presented at appropriate times</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Feedback</strong></th>
<th>Players must receive appropriate feedback at appropriate times</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• players should receive feedback on progress toward their goals</td>
</tr>
<tr>
<td></td>
<td>• players should receive immediate feedback on their actions</td>
</tr>
<tr>
<td></td>
<td>• players should always know their status or score</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Immersion</strong></th>
<th>Players should experience deep but effortless involvement in the game</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• players should become less aware of their surroundings</td>
</tr>
<tr>
<td></td>
<td>• players should become less self-aware and less worried about everyday life or self</td>
</tr>
<tr>
<td></td>
<td>• players should experience an altered sense of time</td>
</tr>
<tr>
<td></td>
<td>• players should feel emotionally involved in the game</td>
</tr>
<tr>
<td></td>
<td>• players should feel viscerally involved in the game</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Social Interaction</strong></th>
<th>Games should support and create opportunities for social interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• games should support competition and cooperation between players</td>
</tr>
<tr>
<td></td>
<td>• games should support social interaction between players (chat, etc.)</td>
</tr>
<tr>
<td></td>
<td>• games should support social communities inside and outside the game</td>
</tr>
</tbody>
</table>
2.2 Platform Games

Platform games, or platformers, were introduced in the early 1980s and constitutes one of the oldest genres of video games. A platformer is a game where the player controls an avatar to navigate through an environment consisting of platforms and obstacles to achieve a goal. [7]

Initially platform games allowed gameplay movement in two dimensional environments, called 2D platform games. Successors have adapted the genre to allow gameplay in three dimensional environments, known as 3D platform games. 3D platform games however, have not been as popular as 2D platform games [8]. Adams believes that this can be explained by the increasing complexity using the controls when introducing an additional dimension [8]. Furthermore, hybrid platform games have been developed by mixing features from 2D and 3D platform games. These games either rely on 2D graphics in 3D environments or on 3D graphics in 2D environments. According to Boutros the platform game genre had 15% share of the gaming market in 1998 and even more at the peak of its popularity; however, in 2002 the marked share had reduced to 2% [9].

2D platform games are usually divided into two categories, single screen platformers and side or vertical scrolling platformers. Single screen platformers use a static camera resulting in gameplay where the player has to complete certain objectives on a single screen. After level completion, the player character is moved to a new single screen level with new objectives. In scrolling platform games the camera follows the player character through a level, typically when the character approaches the screen boarders. This technique results in a scrolling effect of the environment and enables possibility for larger game levels. [7]

This section addresses the structure and design principles of platform games and game levels, focusing on aspects and features enhancing player enjoyment in platform games. The section is divided into ‘game play & game mechanics’ and ‘game interfaces’ as identified by Clanton for addressing the different areas of game design in a structural manner [3]. There is also a section covering important aspects concerning graphics and audio.
2.2.1 Game Play and Game Mechanics

Game play and game mechanics comprises level components, character movement and the structure of a game, including challenges and goals and excluding features such as music, graphical art and game interface. A common separation of the challenge in game play is action-based game play; also called dexterity based game play, and puzzle-based game play. Dexterity-based platform games are considered platform games that focus on the difficulty of traversing a level using dexterity or skill in contrast to platform games concentrated on introducing puzzle aspects. [9]

One approach of introducing puzzle aspects in games is to let the player chose between characters with different abilities to find the most appropriate character for a certain task [10]. Usually platform games consist of both action based game play and puzzle-based game play. In this case, research on previous popular platform games has proven that action based gameplay is favored over puzzle-based game play in initial stages of the game [9].

The following six subjects are covered in this section:

1. Platform game components and level structure
2. Player experience and fun in 2D platform games
3. Procedural level generation
4. Challenge and rewards
5. Player character movement
6. Common tasks and obstacles

2.2.1.1 Platform Game Components and Level Structure

Levels in platform games are composed by several common elements which are structured in various fashions. Smith, Cha & Whitehead have provided a framework for modelling and analyzing the design of platformer levels [10]. The framework focuses on the structure of levels and excludes the visual representation [10].

The elements of platform game levels were identified and mapped to different categories, also called components, in consideration to their role in the levels. Following five components were presented: Platforms, Obstacles, Movement Aids, Collectible Items and Triggers. Each element belongs to either one or several categories. Platforms are defined
as the elements that are used to traverse the level and the properties of the platforms may vary. Usual properties are size, slope and friction. They may be stationary or in motion as well as constant or temporal. Obstacles are elements that invoke danger and are able to harm the player character. They can be static or moving, destroyable or imperishable. Even though bottomless pits that instantly perish the player character are not objects per say, they were placed in this category. Movement aids are considered objects that assist the player in traversing the level, the effect granted from a movement aid must last over a limited amount of time. Collectible items are objects that yield rewards, such as coins or power-ups. The triggers category includes objects that trigger events such as altering of the game environment. The effect may be constant or temporal. [10]

The part of the framework regarding composition of the components focuses on rhythm pacing and rhythm groups [10]. Rhythmic patterns are considered a central aspect of platform game level design that, according to Nicollet considerably contributes to the grade of challenge in platformer levels [11]. The occurrence of rhythm is in the combination of the way the player controls the character, for example when performing three timed jumps in a sequence [10].

Smith, Cha & Whitehead divided the platformer level structure into portals, cells, rhythm groups and components (see Figure 2.1). Cells were considered the highest subdivision of a level representing sections of linear gameplay. A level may contain several cells which are linked by portals. Portals are defined as the transition area between two or more cells. A rhythm group consists of a relatively short sequence of components that makes out one challenging part of a level. The rhythm groups are separated by sections where the player can rest where after the rhythm changes and a new challenging part begins. A specific rhythm group may occur several times in a level, possibly with changes in the graphical representation. [10]

### 2.2.1.2 Player Experience and Fun in 2D Platform Games

Pedersen, Togelius & Yannakakis have examined how level design parameters in platform games impacts player emotions in order to computationally model player experiences from gameplay interaction. A modified version of Infinite Mario Bros was used in their experiments to find relationships between level design parameters, player gameplay characteristics and player experience. [12]

Several versions of game levels were built by modifying various controllable features used in the level construction algorithm. The parameters for the controllable features were: number of gaps, average width of gaps, spatial diversity of gaps and number of direction
Chapter 2. Related Work

Figure 2.1: Conceptual model of the framework presented in [10].

switches the player character had to perform to complete the level. The different levels were made available on the web where test-subjects could play the levels and answer a questionnaire concerning their emotional preferences according to fun, challenge and frustration for each level. Also, statistical features such as level completion time, time spent on various tasks, information on collected items, killed enemies and information on how the player died were logged during the gameplay sessions. [12]

By analyzing the collected data from the experiment several statistically significant correlations between expressed preferences, gameplay features and controllable game level features could be observed (see Table 2.3). In conclusion, players seemed to experience the most amount of fun in fast paced games with quick progression. Killing many enemies and collecting a high amount of rewards also elevated the experience significantly. Kicking turtle shells was the most statistically significant feature correlated with fun which demonstrated that the triggering of complex chain events tended to be an appreciated feature. Challenge was found to especially depend on gap width and gap entropy and none of the features correlated with fun were also correlated with challenge. Frustration seemed to rise when players were not able to complete a level, died by falling into gaps and in game level designs where the player character had to spend much time standing still. [12]

One way to minimize the risk of frustrating players is the “almost there” effect discussed by Nicollet. This effect can be obtained by introducing a moderate amount of checkpoints in the game levels; if a player dying due to a difficult obstacle is able to see a checkpoint
### Related Work

1. Number of times the player kicked a turtle shell
2. Proportion of coin blocks that were “pressed” (jumped at from below)
3. Proportion of opponents that were killed
4. Number of times the run button was pressed
5. Proportion of time spent moving left
6. Number of enemies killed minus times died
7. Proportion of time spent running

#### Fun

1. Whether the level was completed
2. Proportion of power-up blocks pressed
3. Proportion of Mario deaths that were due to falling into a gap
4. Number of times Mario died from falling into a gap
5. Jump difficulty
6. Average width of gaps
7. Number of times Mario ducked
8. Proportion of time spent in the last life
9. Proportion of coin blocks that were pressed
10. Number of gaps

#### Challenge

1. Whether the level was completed
2. Proportion of power-up blocks pressed
3. Proportion of coin blocks that were “pressed” (jumped at from below)
4. Number of times Mario died from falling into a gap
5. Proportion of Mario deaths that were due to falling into a gap
6. Proportion of collected items
7. Time spent standing still
8. Proportion of killed opponents that were killed with fireballs
9. Proportion of time spent in the last life
10. Proportion of coins collected

#### Frustration

1. Proportion of coins collected

| Table 2.3: Top ten statistically significant correlation coefficients between reported emotions and extracted features in decreasing order as described in [12]. White cells corresponds to positive relations while red cells indicates negatives relations. Observe that only seven statistically significant correlations related to 'fun' were found. |

at a reasonable range past the obstacle, he will get the feeling that he is close to reaching the sub-goal and happily retry the challenging section instead of getting frustrated. [11]

In subsequent research based on the research in [12], Shaker, Yannakakis & Toglius further investigated player experience in the Infinite Mario Bros platform game [13]. Results indicated that features like number of enemies and number of gaps were important for evoking player emotion irrespectively of the objects placement in the game level [13]. In addition, the number of power-ups had a remarkable impact on the player engagement when the objects occurred in the second half of the level [13].
2.2.1.3 Procedural Level Generation

Procedural level generation or procedural content generation in the area of 2D platform games refers to the automatic or programmatic generation of content in game levels. This can be either fitting together pre-designed chunks of levels or completely generate levels based on various parameters [14, 15].

The generation can also be performed either to predetermine levels that may be saved and used later or in real time, i.e. when the game is running. The degree of automatic generation may vary but the main reason for using procedural content generation is to increase replayability. With this method, a large amount of levels can be generated with little effort from the designer. A challenging part with the approach is however to create levels that obtain the same amount of quality as hand-crafted levels, mainly in respect to variation. [16]

Fisher has proposed three requirements that procedural level generation algorithms should adhere to. The first one state that the generated levels must be feasible, meaning the levels must be possible to complete. A way to ensure that the created levels are beatable is to create an AI player that tests every generated level. The second requirement says that the levels must be interesting. For example, the level displayed in Figure 2.2 is feasible, but not interesting. The level is monotonic and possibly too short. Ensuring this requirement is more complex than ensuring whether a level is feasible. The third and last requirement says that the generated levels must be sufficiently challenging. [16]

![Figure 2.2: A screenshot of an 'uninteresting' procedurally generated platformer level from [16].](image)
2.2.1.4 Challenge and Rewards

Challenge and reward design are closely coupled and constitutes one of the most important domains of platform game design.

Reward systems have existed in platform games since the very beginning of the genre. Common rewards among some of the oldest and most popular platform games are financial icons, such as coins in Super Mario Bros or rings in Sonic the Hedgehog, extra-lives which increases the number of times a player character may die, power-ups that enhances the player in various fashions and hit-point-up objects which increases the amount of damage that the character can be exposed to without dying. Permanent upgrades were also found in several of the investigated games, though not with the same extent. [9]

Another reward is so called quota-tokens. A quota token is an item that is part of a set and after collecting the complete set a reward is granted. Examples of quota-token rewards are: gained access to special areas, bonus game unlocks, character unlocks and gained access of new abilities. The quota tokens are typically found in hidden areas or in otherwise hard to reach areas. Recently, quota tokens have increased in popularity and are being used frequently in new games, partly because its positive influence on the replay-value of the game. For the same reason, newer games also tend to reward exploration over gameplay skill. Further, newer platform games have been observed to be greedier with rewards overall. [9]

Comparison between popular 2D platform games have shown that three to five power-ups per level seems to be a good balance and introducing power-ups early in the game is likely to immediately increase the player enjoyment of the game. Moreover are unique power-up systems a good way to create further distinction to games. [9]

Nicollet writes about difficulty in dexterity-based platform games and emphasizes the importance of balancing the level of difficulty in the game genre for creating a challenging yet entertaining platform game. A number of rules are set out intended to give players and developers a better understanding of difficulty in platform game level design. [11]

One rule state that surprise is not difficulty. Surprising the player for example by introducing an unknown obstacle towards the end of a level will often result in frustrating the player, hence surprises should be avoided. This means that all features should be introduced before being used to create difficulty, preferably alone and with variations. Some of the rules are more concerned with how to increase game level difficulty. One way is to increase the loss when a player makes mistakes. Loss of time, loss of collectibles and loss of health-points are some of the parameters that can be modified to accomplish this. [11]
Other rules related to increasing difficulty presented by Niccollet are [11]:

- Reducing the time available to go past an obstacle increases the difficulty.
- The longer a player must act without pause, the greater the difficulty.
- Increasing the required precision to go past an obstacle increases the difficulty.
- Long and complex patterns increase the difficulty.
- The further the obstacle, the harder the prediction becomes, and the difficulty increases.

A common design choice when incorporating challenge in platform games is to add secondary objectives. Secondary objectives are optional challenges that the player can choose whether or not to pursue. The use of secondary objectives may result a game with more freedom and flexibility where players can try to accomplish these tasks when not focusing on the main goal. It may also enhance the games’ replayability value. However, research conducted with data from over 27,000 players has proved that secondary objectives may harm the retention of many players decreasing the time they are willing to play the game. It was discovered that secondary objectives increased the entertainment value of the game for some players with high level of engagement, while the objectives affected the average player in a negative manner since the average player often felt obligated to pursue the secondary objectives instead of ignoring them. Thus when used, secondary objectives should be implemented in such way that it reinforces the primary goal of the game and profound testing is necessary to ensure that the objectives don’t have a negative impact on the players. [17]

2.2.1.5 Player Character Movement

The player character often possesses abilities to run, jump, shoot and crouch [10]. Adams points out that most platform games uses unrealistic physics, e.g. the player can change the direction of a character mid-air [8]. In some games the character may also perform wall jumping and so called double jumps, the latter means that the player may jump one time in the air after performing a jump from the ground [10].

Boutros states that a high number of character actions available in a platform game will increase the players’ feelings of freedom and allows for more potential ways to interact with the game. Another important factor regarding player character movements is that a constant flow should exist in the game play meaning that there should be no or short interruptions between character actions. [9]
2.2.1.6 Common Tasks and Obstacles

Some tasks and obstacles are common for several of the most popular 2D platform games. This section describes some widely existing tasks and categories of obstacles within platformer levels.

It is important for games to engage the players early on and to maintain engagement and attention throughout the game. For this reason and due to the importance of getting player used to the game world rules at an early stage, platform games generally introduce a large variety of tasks early on. Boutros have established a set of tasks that popular platform games tend to introduce within the first ten minutes of gameplay. The six most applied tasks are: collecting finance, find power-ups, jump over kill-zones, destroy containers, earn bonus quota tokens and jump over obstacles moving horizontally on the floor. Other tasks are: accessing heard to reach areas, killing floor based enemies, using power-ups to reach certain areas, hit enemies that can only be harmed at specific periods, collecting enough finance to access bonus stages and to bounce on enemies in a sequence without touching the ground. [9]

Nicollet describes some reoccurring categories of obstacles in platform games. One category is the pits, also called bottomless kill-zones, which will instantly kill or damage the player character when not avoided. The bottomless kill-zones should be presented in a way that makes it obvious that not avoiding them will harm the player. Static obstacles, such as spikes in Sonic the Hedgehog 2, may be included in this category. Pendulums are another category of common obstacles which is defined by obstacles that constantly moves with regularity in a somewhat predictable manner. The movement of the pendulum is unaffected by the player, one examples is the jumping fireballs in Sonic the Hedgehog (see Figure 2.3). Another category is the unstable floor. This is platforms that can become hostile or trigger traps once the character steps on them. Examples are floating bricks in Crash Bandicoot that will fall down into harmful water moments after being stepped on or platforms that will start burning in Sonic the Hedgehog (see Figure 2.4). The occurrence of these obstacles should be made aware to the player and should beneficially be reset after a while in case the player chooses to jump back to the stable platforms from which he came instead of moving forward. [11]

Obstacles introduced at an early stage in platform game levels often consist of floor based enemies, static obstacles and bottomless kill zones with the enemies often having part of their body protected. A usual design strategy in platform games is also to surround the level area with water or empty space that can instantly kill the player character. [9]
2.2.2 Game Interface

The game user interface turns the player input into actions in the game [8]. Platform games exist on a wide variety of platforms, from personal computers to home video game consoles and handheld game consoles as well as on mobile phones [18]. Hence several types of control schemes have been developed and are used in platform games.

A common factor for many of the most popular platform games is that relatively few buttons are being used for character movement and actions. In the study by Boutros, the average number of buttons used was three. He also stated that the accessibility of games increases the less the players need to learn mentally and physically. [9]
Adams has provided some principles to keep in mind regarding game interface and user interface when developing games in general; however, most of these principles also applies to the platform game genre [8]. Following is a summary of some of the general principles presented by Adams:

- **Be consistent.** This applies to both aesthetic and functional issues; the game should be stylistically as well as operationally consistent.

- **Give good feedback.** When the player interacts with the game, he expects the game to react immediately.

- **Remember that the player is the one in control.** Players want to feel in charge of the game, at least in regard to control their avatars.

- **Limit the number of steps required to take an action.** Set a maximum of three controller-button presses to initiate any special move.

- **Minimize physical stress.** Video games have a tendency to cause physical stress injuries. Assign common and rapid actions to the most easily accessible controller buttons.

- **Don’t strain the player’s short time memory.** Don’t require the player to remember too many things at once; provide a way for him to look up information that he needs. Display information that he needs constantly in permanent feedback elements on the screen.

- **Group related screen-based controls and feedback mechanisms on the screen.** That way, the player can take in the information he needs in a single glance.

- **Provide shortcuts for experienced players.** Once players become experienced with the game, they won’t want to go through multiple layers of menus to find the command they need. Provide shortcut keys to perform the most commonly used actions from the game menus.

### 2.2.3 Graphics

Graphics and music are vital areas in platform game design for producing enjoyable game experiences. Historical events have demonstrated that the old concept “graphics doesn’t matter” is a falsehood; however, the graphics and the game mechanics goes hand in hand and thus extraordinary graphics alone is not enough for creating a good game [9].
Chapter 2. Related Work

The size of the player character on the screen is an important graphical aspect. The character should be clearly visible, but at the same time as much as possible of the environment should be displayed. By comparing various popular 2D platform games it was established that a character height of one fifth to one sixth of the screen height was a balanced character size. Further, characters with a large head to body ratio is preferred for producing good character distinction and memorable characters. [9]

Boutros also address the importance of platform games having a theme that allows for an environment with intuitive and colorful surroundings, for example levels that takes place in a tropical jungle [9].

2.2.4 Audio Elements

Music, ambient sound and sound effects are some of the most important topics when it comes to audio elements. A general rule regarding audio is that all games should offer an option for adjusting the volume of the music and other audio effects independently. In this way, players can choose to turn off music and keep sound effect and vice versa. Sound effects are sound that corresponds to actions and events. A common use of sound effects is to increase the player’s attention, for example to warn about upcoming dangers. It can also be applied to enhance the user interface by providing feedback when interacting with menu components. Ambient sound is an audio element that is being played to give feedback about the environment in which the game takes place, for example animal calls in a jungle environment. These kinds of sounds should be very subtle; especially for games that requires a high amount of concentration. [8]

Music in games may enhance the playing experience and contributes to establishing the pace of the game [8]. The music should be in harmony with the game theme and background music with wide dynamic range should be avoided for making repetition more discrete [8]. Nicollet discusses how music can be applied in platform games to affect the player [11]. Increasing the tempo of the music just before a challenging section of a game level will affect the player to move forward in a faster pace resulting in increased probability for the player to make mistakes, hence making the specific section even more challenging [11].

2.3 Motion Based Games

Motion based games are games that uses the physical movements of the player for interacting with the game. These kinds of games are becoming increasingly popular and various techniques exist for allowing players to interact with games using motion.
The Wii-console developed by Nintendo created a huge upswing for motion based games in the gaming industry when released in 2006 [18]. The console came with a game controller, called the Wii-remote, which utilized accelerometers and Infra-Red transmitters to enable motion based interaction in games [18]. Sony Eytoy, Microsoft Kinect and Sony Playstation Move are further examples of sensor technology incorporated in gaming consoles. Similar techniques also exist for mobile phones and for personal computers [19].

2.3.1 Motion Based Interactions

Motion based interaction is claimed to be a more intuitive form of interaction than using the standard desktop computer controls, i.e. mouse and keyboard. In this concept, intuitive means that users of a system unconsciously or automatically enact appropriate inputs to perform a certain action in games. [20]

In a study by Bianchi-Berthouze, Kim & Patel it was detected that body movement in video games increases the engagement level of the players [21]. Further research in the area stated that motion based games also increases the reported arousal level of the players [22]. By comparing Wii-games with varying amount of motion required to control them, it was established that more movement increased the energy level of the players after game play [22]. However, no statistically significant correlation was found between amount of movement and reported level of fun [23]. Additionally, motion based games tend to further increase the social connection among player playing together in comparison to non-motion based games [22].

Embodied interaction is an approach to interaction design that focuses on understanding and incorporating peoples’ natural body movements into controlling computational systems. Research in the area have proved that using conceptual metaphors derived from image schemas and operating outside of conscious awareness, invoke a systematic and predictable interaction scheme which results in a more enjoyable playing experience. Conceptual metaphors refer to the understanding of one idea in terms of another. Image schemas are formed from bodily interactions and are used within the cognitive process to establish patterns. [24]

Motion based interaction systems must be designed to allow for ergonomic gestures and to prevent gestures that may cause damage to the player [22]. Isbister emphasizes that design principles and guidelines for motion based interaction systems should be focused on how the required movements make the players feel physically [22]. Isbister & Mueller have presented a set of guidelines regarding motion based interactions that may be used by game developers to increase the chance of creating successful motion based games.
The guidelines were examined and refined by several game designers experienced in the area of movement based games [19, 25]. Table 2.4, 2.5 and 2.6 features the ten guidelines separated in three clusters [19, 25].
# First Guideline Cluster: Movement Requires Special Feedback

<table>
<thead>
<tr>
<th><strong>Guidelines</strong></th>
<th><strong>Strategies for designer &amp; DOs and DON’Ts</strong></th>
</tr>
</thead>
</table>
| **Embrace Ambiguity**  
Ambiguity in movement-based games arises from the fact that no two movements are the same, and most sensor data is messy. Trying to force precision may only frustrate the player, and make the limitations of the sensor obvious in a very un-fun way. So instead of trying to remove this ambiguity, work with it: players enjoy surfing uncertainty and trying to figure out optimal strategies in a somewhat messy system. | • Get to know the limits of your sensors, and use these limits as a design resource.  
• Construct the player’s action in a way that gives room for sensor error without drawing attention to it.  
• Avoid game mechanics that require precise control.  

DO use the ambiguity of movement and sensor data to enhance the game.  
DON’T use buttons during the early development phase (even if it’s easier), as you will miss the opportunities arising from dealing with ambiguity. |
| **Celebrate Movement Articulation**  
In button-press games players get feedback if and when they press a button. With movement-based games, it is not just if and when, but also how movement is performed. Also, you are not always performing movements to achieve an outcome. Sometimes movement can be enjoyable on its own (whereas pressing a button is not usually a noteworthy pleasure for the player). Therefore celebrate the joy of movement and its articulation by providing players with feedback on the quality of their movement. This feedback has to be instantaneous, so that players can improve their movement articulation moment-by-moment. | • You need not judge articulation, you can just provide feedback by highlighting players’ articulation to allow them to reflect on and learn from it by themselves.  

DO provide feedback if and when movement occurred, but also on how.  
DON’T worry about judging the how, players can figure it out themselves as long as they get feedback. |
### Consider Movement’s Cognitive Load
Developing movement skill requires not only bodily, but also cognitive attention, with is a limited resource. Initially, players need to focus on learning a new movement (so focus the feedback on this). Once they are better at the movement, they can devote more attention to more complex and nuanced forms of feedback.

- Start by providing feedback on the movement itself, without too much worrying about scores, multipliers etc.
- Provide several forms of feedback, but do not require players to engage all of them: better to let players choose which ones to engage based on their cognitive abilities, and shift their attention as mastery grows.

**DO** reduce cognitive complexity during movement: for example, if your player can usually remember three rules, as soon as she/he moves, she/he will only remember one.

**DON’T** forget once players learn new movements, they may need to re-learn old ones as they integrate new skills.

### Focus on the Body
Focus on the body, not just the screen, when designing player feedback. In movement-based games, the body is a major focus of attention: audiences enjoy watching moving bodies, and players listen to their own bodies via proprioception. Do not distract players from this focus on the body by drawing too much attention to the screen.

- Think past screen-based feedback. You can use audio and haptics, as well as other players to offer feedback.

**DO** start imagining your game without a screen.

**DON’T** forget that for players who feel self conscious or reluctant to move, diverting attention away from the body might be beneficial to reduce the barrier to play.

<table>
<thead>
<tr>
<th><strong>Consider Movement’s Cognitive Load</strong></th>
<th><strong>Focus on the Body</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Developing movement skill requires not only bodily, but also cognitive attention, with is a limited resource. Initially, players need to focus on learning a new movement (so focus the feedback on this). Once they are better at the movement, they can devote more attention to more complex and nuanced forms of feedback.</td>
<td>Focus on the body, not just the screen, when designing player feedback. In movement-based games, the body is a major focus of attention: audiences enjoy watching moving bodies, and players listen to their own bodies via proprioception. Do not distract players from this focus on the body by drawing too much attention to the screen.</td>
</tr>
<tr>
<td><strong>DO</strong> reduce cognitive complexity during movement: for example, if your player can usually remember three rules, as soon as she/he moves, she/he will only remember one. <strong>DON’T</strong> forget once players learn new movements, they may need to re-learn old ones as they integrate new skills.</td>
<td><strong>DO</strong> start imagining your game without a screen. <strong>DON’T</strong> forget that for players who feel self conscious or reluctant to move, diverting attention away from the body might be beneficial to reduce the barrier to play.</td>
</tr>
</tbody>
</table>

Table 2.4: First cluster of the guidelines regarding movement based interactions presented in [19, 25].
### Guidelines Cluster Two: Movement Leads to Bodily Challenges

<table>
<thead>
<tr>
<th>Guidelines</th>
<th>Strategies for designer &amp; DOs and DON’Ts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intend Fatigue</strong>&lt;br&gt; Movement results in fatigue. On the one hand, it can be a welcomed challenge for players if they have to manage this fatigue (for example in endurance sports), on the other hand, fatigue can negatively affect engagement. Therefore, intend fatigue when using it as a game challenge, but avoid it when it is not part of the game.</td>
<td>• Minimize chances of fatigue by creating short game cycles.&lt;br&gt; • Minimize chances of fatigue by varying movements.&lt;br&gt; • Distract players from fatigue, e.g. through music.&lt;br&gt; • DO use the management of fatigue as a game mechanic.&lt;br&gt; DON’T assume players know how to manage fatigue, support them in figuring it out.</td>
</tr>
<tr>
<td><strong>Exploit Risk</strong>&lt;br&gt; Exploit physical risk sensibly. Movement, especially in everyday indoor environments, has an inherent sense of risk associated with it: there is risk of injury, risk of breaking furniture, risk of hitting another person. However, with risk also comes a sense of thrill, which can contribute positively to the game experience</td>
<td>• Make players aware they are engaging in risky activity.&lt;br&gt; • Consider the environment when exploiting physical risk.&lt;br&gt; • Let players’ movements interfere with each other to facilitate body contact, which has physical risk associated with it.&lt;br&gt; • DO put the player's safety first.&lt;br&gt; DON’T assume players will be fully aware of any emerging physical risks, as they might be distracted by engaging play.</td>
</tr>
<tr>
<td><strong>Map Imaginatively</strong>&lt;br&gt; Map movements in imaginative ways. The computer allows mapping movements in many imaginative ways, in particular in ways that are not possible in real life, offering players fantasy-fuelled opportunities they do not have otherwise. Mapping does not need to be literal or slavishly true-to-life.</td>
<td>• Map movement in a non-linear fashion, for example in a tennis fantasy game a weak forehand movement results in a strong hit.&lt;br&gt; • Add additional virtual movement to mapped movement.&lt;br&gt; • Engage “avateering”: make the player’s movements look better than they really are.&lt;br&gt; • DO engage your creativity in the mapping process.&lt;br&gt; DON’T use this guideline if you want to simulate a realworld sports experience, such as designing a golf simulator.</td>
</tr>
</tbody>
</table>
Table 2.5: Second cluster of the guidelines regarding movement based interactions presented in [19, 25].

<table>
<thead>
<tr>
<th>Guidelines</th>
<th>Strategies for designer &amp; DOs and DON’Ts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Highlight Rhythm</strong></td>
<td>• Play music for players to help them identify a beat.</td>
</tr>
<tr>
<td>Help players identify rhythm in their movements. Movement is rhythmic. Movement can be rhythmic on an individual action level (tennis swing), but also the overall movement experience often follows a rhythm of high and low-intensity actions. Movement becomes easier with a beat, so support players in identifying a rhythm to their movements.</td>
<td>• Visualize previous and upcoming movements so players can identify a rhythm in their movements.</td>
</tr>
<tr>
<td></td>
<td>• Think of haptics not only as a feedback mechanism for action, but also as a rhythm aid.</td>
</tr>
<tr>
<td></td>
<td>DO see movement in games not just as a string of independent actions, but as a sequence of rhythmic actions that, with a beat, get easier. DON’T forget that engaging competitive gameplay can emerge when allowing players to try to throw their opponents off their beat.</td>
</tr>
<tr>
<td><strong>Support Self-Expression</strong></td>
<td>• Allow players to perform different kinds of movements to achieve the same outcome.</td>
</tr>
<tr>
<td>Support players in expressing themselves using their bodies. We communicate a lot about ourselves in how we move. Thus playing a movement-based game is always a form of self-expression, especially with other people around us. Take advantage of this to increase fun for players.</td>
<td>• Encourage players to try out these different movements.</td>
</tr>
<tr>
<td></td>
<td>• Celebrate self-expression by showing players the result of their self-expression, for example in forms of photos of their movements as trophies.</td>
</tr>
<tr>
<td></td>
<td>DO see movement as a form of self-expression that can make your game more fun. DON’T forget self-expression is not only concerned with the player, but also with other players and any audience.</td>
</tr>
</tbody>
</table>
### Facilitate Social Fun

Facilitate social fun by making movement a social experience. Moving with others is fun. Movement is typically visible to others and easily becomes a performance, whether we intend it or not. Therefore, design for multi-player, including other players and an audience.

- If you plan to design both multi-player and single-player modes for your game, consider starting with multiplayer.
- Make the game a spectacle: encourage movements that are, by nature, a spectacle others enjoy watching.
- Turn bystanders into players: allow the audience to easily join the game.
- Make the game easy to learn by observing, so that spectators figure out what is going on quickly and want to try.

DO engage other players and audiences by turning the movement into a performance.

DON’T forget that movement in spaces where others do not know that there is a game going on, such as public spaces, might create socially awkward situations.

<table>
<thead>
<tr>
<th>Table 2.6: Third cluster of the guidelines regarding movement based interactions presented in [19, 25].</th>
</tr>
</thead>
</table>

### 2.3.2 Game Play & Game Mechanics

Using the body and its movements for interacting with machines introduces new opportunities when it comes to game design, but the interaction scheme also leads to new problems. Motion controllers are for example subject to recognition errors and delays and game developers must take these factors in mind when designing motion based games [19].

Sinclair, Hingston & Masek have examined how to design exergames which is a mixture of video games and exercising equipment. They state that when trying to get players to reach a state of flow in non-movement based games, focus generally lies on finding a balance between the skill level of the players and the challenge of the game levels. This is often accomplished by developers performing extensive testing and results in levels with predefined grades of challenge. However in movement-based games, the daily fitness level of the player varies affecting the perceived challenge. Hence, these kinds of games must take fitness level into consideration. This can be performed my monitoring the players’ current skill level and use this data to dynamically modify the difficulty of the levels. [26]

In research focused on heightening social and emotional engagement and enjoyment in motion based games, a series of popular and well-reviewed Nintendo Wii games were
analyzed to detect common movement mechanics and design choices [27]. For character-
ing movement based mechanics Laban’s Movement Analysis system was applied with
focus on the three dimensions of effort where movements could be described as: direct
or indirect, strong or light, and bound or free [28].

One of the key findings was that all of the examined games used kinesthetic mimicry
for incorporating movement mechanics. That is, all movements were partially patterned
after physical movements already known by the players. Some of these kinesthetic
mimicries also had higher positive impact on the player experience than others. It was
found that movement mechanics with looser criteria for recognizing movements and
allowed for player improvisation was preferred, even though the resulting movements
may not exactly imitate the corresponding game action. The Wii remote was often used
to mimic objects from the real world, such as tennis rackets or swords. The fact that
the actual physical objects was not used in the game play, but the player had to imagine
these objects, were described not to affect the enjoyment of the mechanic. [27]

Another finding was that some games used combinations of movement and button presses
on the Wii remote in a way that negatively impacted the player engagement. This was
combinations related to precision where complex patterns had to be applied to execute
certain actions. For example, in Star Wars: The Force Unleashed, the player can wave
with the light sabre by moving the Wii remote or throw the light sabre on enemies by
simultaneously pressing a button and moving the Wii remote. In this case, throwing
the light sabre was considered less compelling and reduced the engagement level of the
players. Other games however, successfully applied movement and button combinations
in a way that felt natural to the players. One example of this was found in Boom Blox,
where blocks could be pulled by first pressing a button for grasping a box and then
moving the Wii remote backwards for pulling it. In summary, intermix of movement
and button presses is an important design aspect in the crafting of movement mechanics.
[27]

It was also detected that the majority of the analyzed games rarely applied movement
mechanics that enforced whole-body movement. Instead, most games included mechan-
ics that let the player be stationary below waist and only required for torso and arms to
be moved. However, games that resulted in whole-body movement for interacting, in-
stead of so called piecemeal motion, elicited more engagement and was considered more
enjoyable for both players and observers. Regarding social interaction, it was pointed
out that cooperative game mechanics in motion based games seemed underdeveloped
and was an area with many opportunities for improvements. [27]
Chapter 3

Method

The basic layout of the project was a qualitative, exploratory empirical research con-
ducted by the means of case study research as described in [29]. The work flow was
guided by a modified version of the agile user experience design process as described by
Unger & Chandler in [30]. It was altered to cope with the various phases of the case
study research methodology. While user experience design revolves around interaction
and collaboration with users the modified approach puts more focus on argumentative
development and expert evaluation. A broad model of the agile process is displayed in
Figure 3.1. The process consist of two phases which are described in more detail below.

---

**Figure 3.1:** An overview of the work process for this thesis. The figure displays a
modified version of the agile project approach presented in [30].
3.1 The Planning Phase

This is the first phase where overall structure and approaches were determined. The case study plan was defined according to [29]. The plan was established in order to provide a clear definition of the studied problem and for making sure that the project evolved in the right direction, making sure that studied areas provided useful information with respect to the research question.

Preparation for data collection and an extensive qualitative literature based study was conducted, covering industrial and scientific literature in the areas of game design, 2D platform games, motion based games and motion based interactions. Most of the data from the literature study was collected from electronic sources. The most relevant information was analyzed and a mind map was created in order to structure and give an overview of the work, in addition various game heuristics and guidelines were established.

3.2 The Iterative Phase

The iterative phase covers the primary development of the product. It consisted of three iterations, the time spent on each iteration depended on the time it took to complete the created tasks for each iteration. The project was limited to three iterations due to time limitations. Several tasks were included in each iteration in order to implement many features over a relatively short period of time. With more iterations over the same amount of time, more time would have been devoted on analyzing and evaluation which was not considered optimal for the purpose of this project. Each iteration contained four steps in the following order:

1. Analyze
2. Design
3. Develop
4. Evaluate

In the analyze step the results from the evaluation step in the previous iteration were examined. New tasks were created based on the information and already existing features that did not comply with the interaction interface or the game heuristics were altered or deleted. For the first iteration however, this step consisted of analyzing and creating tasks based on the findings in the literature, feedback from the customer and on own experiences after play testing the already existing product.
In the second step the tasks were evolved into more detailed specifications based on the current design of the game. Implementation and refining of the solutions were carried out in the third step. The implementation was based on the desktop and mobile HTML5 game framework Phaser and most of the code was written in Javascript.

In the last step the new product were evaluated by performing usability inspections based on the heuristics found in the literature and by analyzing the insights generated from coding and looking for patterns. The evaluation process also involved play testing carried out by me and the customer with focus on eliciting features that coped well with the interaction scheme and features that had to be modified or removed. In this case, the customer possessed plenty of expertise in the area of game design and motion interaction. Furthermore, some informal usability tests with smaller test groups were conducted during the project.
Chapter 4

Results

This chapter presents the results of each of the phases described in method, chapter 3. Since this thesis was based on an existing game project, the original state of the project is important to consider when observing the results. Description of the initial game is found in section 1.5.

4.1 The Planning Phase

Table 4.1 describes the elements of the defined case study plan.

The heuristics, models and guidelines that were retrieved from the literature study and constitutes important parts of the evaluation process of the various implementation iterations can be found in Table 2.2, Table 2.4, Table 2.5 and Table 2.6. The tables will henceforth, when referred to in the evaluation section of the iterations, be denominated as Game flow model, Motion guide cluster 1, Motion guide cluster 2 and Motion guide cluster 1 accordingly.

<table>
<thead>
<tr>
<th>Case Study Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective</td>
</tr>
<tr>
<td>The case</td>
</tr>
<tr>
<td>Frame of reference</td>
</tr>
<tr>
<td>Research question</td>
</tr>
<tr>
<td>Methods and selection strategy</td>
</tr>
</tbody>
</table>

Table 4.1: Description of the elements from the case study plan.
4.2 The Iterative Phase

This section describes the tasks that were created and implemented in each iteration. The description includes explanation of the tasks as well as declaration of data and arguments leading to the creation of the tasks. Results from the various tests in each evaluation step are also presented. Relevant heuristics and guidelines were graded on a scale from zero to five depending how well the game conformed with them, a grade of zero implied 'not at all' while a grade of five implied 'very much'. In addition, each iteration section also features a table displaying all implemented tasks for each iteration in a comprehensive manner.

The tasks created in the first stage of each iteration phase were often based on features from regular 2D platform games, subsequent usability tests were then conducted for establishing if those features did fit with the interaction scheme or had to be modified.

4.2.1 Iteration 1

The first iteration primarily included implementation of basic features and components, both from games in general and from 2D platform games. Much of the focus lied on creating the fundamental structure of the game and making it playable.

4.2.1.1 Tasks

A system for player character health was introduced along with the ability to damage and get damaged by future obstacles and monsters. Health values for monsters were also implemented. The score user interface positioned on the top of the game screen that previously only displayed the score for collected stars (see Figure 1.1) was updated with a health bar representing the health of the player character. By default, the player character was assigned a health value of 100. When damaged by a monster the health value was decreased with the damage amount specified by the monster object type and when inflicting damage to a monster the health value of the monster was reduced accordingly. When the character was damaged a timer was triggered that made the character invulnerable for a specific period of time. Monsters were designed to inflict damage to the player character on collision, the only exception was when the player collided on the top of the monster; in this case the monster was damaged. This typically occurs when the player jumps on the head of the monster. Furthermore, a checkpoint component was implemented with the expected effect of minimizing player frustration as described in section 2.2.1.2. When the player died, the position of the player was
Figure 4.1: An example of what could be displayed in the level complete state after completing the first level.

reset, either to the position of the spawn point, as described in the next paragraph, or at the last passed checkpoint.

Spawn points and goal components were implemented along with a structure that made it possible to have multiple levels in the game. This included the creation of several levels and a “level complete” state that displayed score and information about the completed level. The state was triggered when the player reached the goal component in a level and contained a button for starting the next level. Figure 4.1 shows an instance of the level complete state after finishing level one. Following list gives a more thorough explanation of the flow of the level structure:

1. When the game starts, a “preloader” state gets started and the first tile map level is loaded to the cache from the corresponding JSON file that was created using the Tiled editor. The cache is where all the externally loaded assets are stored.

2. When the map has been fully loaded, the “level” state that handles the actual creation of the levels, the game play, and all game mechanics is started.

3. The map for the level is loaded from the cache and is used to create the level. Consequently, a player character is created at the position of the invisible spawn
point that was included as an object in the tile map. This is also where all remaining game objects are created and positioned according to the tile map.

4. Once the player reaches the goal, the level and all of its components are destroyed and the level tile map is deleted from the cache to minimize memory usage. This is followed by an event that starts the “level complete” state to which various score parameters from the completed level is passed.

5. When started, the “level complete” state first loads the tile map for the next level to the cache in accordance with the first step. Secondly, the various score parameters from the previous level are presented and a button for starting the next level appears.

6. When the button is activated, the “level” state is once again started and the procedure continues from step three.

Note that this implementation allows for levels with variation in length and structure. Figure 4.2 displays a flow chart of the aforementioned structure.

The size of the player character and the environment was altered based on the information in section 2.2.3. While a character height of one fifth to one sixth of the screen height was considered balanced, the character in this project was set to cover approximately one seventh of the screen height. This was a size that made it easy to distinguish the character and still allowed the player to see a satisfying range of the environment around the character. A comparison of the original and the new size of the player character and the environment are found in Figure 4.3. Game and character mechanics were also altered where reasonable values were set for following parameters:

- **Drag in x-axis**, which specifies how fast the character will reach the velocity of zero when no acceleration is applied. This parameter was adjusted both for situations where the character had contact with the ground and for situations where the character was located in the air, i.e. when not touching the ground.

- **Maximum velocity**, which restricts the velocity of the character. This value was set both for velocity in x-axis and for velocity in y-axis.

- **Jump height**, which determines the amount of velocity applied on the character in y axis when a jump action is performed.

- **Character mass**, specifying the mass of the character.

- **Gravity**, which can be described as the constant force affected on the character in y-axis.
Chapter 4. Results

Figure 4.2: A flow chart displaying the level structure procedure.

Figure 4.3: A comparison of character size and environment size prior and after modification.
Several other game components were introduced in iteration one, including bottomless pits, moving platforms, springboards, movable crates, first aid power-ups and quota tokens. Bottomless pits belong to the obstacle category as described in section 2.2.1.1 and consists of holes or spaces between platforms that instantly kills the player character if not avoided. According to Nicollet, this is one of the most reoccurring types of obstacles in platform games (see section 2.2.1.6). Moving platforms are platforms with constant motion traveling back and forth a path defined by platform stop zones in the Tiled editor, the direction of travel might be either in x- or y-axis. The platforms were designed to have a size of two tiles in width and one tile in height. The velocity of the platform was determined by a property belonging to the object in the Tiled editor.

The spring board is a movement aid that, when stepped onto, ejects the character in an upwards direction with some force in negative or positive x-axis depending on the properties of the spring board. The crates fit into the category of movement aids as well. The character can push the crates around and also jump onto them to reach platforms or other game objects that are otherwise unreachable. They have drag, a defined mass and are affected by the gravity. First aid power-ups are collectible items that increase the health of the player character when collected. Power-ups in general may increase the player engagement (see section 2.2.1.2) and also the player enjoyment of the game (see section 2.2.1.4). Those were the main reasons why power-ups were implemented in this project. In addition, finding power-ups is one of the most common tasks that popular platform games tend to introduce within the first ten minutes of gameplay (see section 2.2.1.6). Quota tokens are described in section 2.2.1.4 and were primarily implemented due to the benefits they impose on the replay value of the game. As with finding power-ups, earning bonus quota tokens is one of the most common tasks introduced within the first ten minutes of gameplay in popular platform games. However, the implementation of the reward system for finding the complete set of quota tokens in a level was not implemented in this iteration. Figure 4.4 shows a screen capture from an example level in the Tiled editor where all of the mentioned game components are used.

The score user interface was updated to show number of quota tokens collected along with a clock symbol that displayed the amount of time spent in the level. Number of collected quota tokens for a certain level, the total time it took to complete the level and number of stars collected in the level were the parameters sent to and displayed in the recently described level complete state at level completion.

Some modifications were made to the motion zones. The existing motion zone layout, see Figure 1.1, often produced situations where the character was unintentionally triggered to jump when the player tried to make the character run resulting in player frustration and lack of control. This was solved by decreasing the height of the jump zones while extending the height of the zones used for making the character run. The standard
Figure 4.4: A screen capture from an experimental level in the Tiled editor with the game components pointed out. Note that the moving platform will have a horizontal motion with turning points at the platform stop zones. The spawn point is visible in the tiled editor but will be invisible in the actual game, this applies for the deadly pits as well.

motion zone layout produced at the end of the first iteration is displayed in Figure 4.5. The amount of motion required to trigger a motion zone was determined by a threshold defined in each motion zone. When the amount of motion applied in a specific zone breached the threshold limit of the zone the zone was considered activated and corresponding movement action was applied to the character. Keyboard buttons were set up to modify the size and the threshold of the motion zones in game time to ease the process of finding appropriate sizes and values for the different types of zones when play testing. The threshold altering keys could also be used by players since the amount of motion needed to be performed by the player to trigger the zones highly depended on the distance from the web camera to the player and also on the light conditions in the room the player was located in. This way, the player could alter the zones to be well adapted for the environment he or she was located in.

A summary of all tasks created and implemented in iteration one is found in Table 4.2.

4.2.1.2 Evaluation

A table displaying the gradings of the evaluated heuristics and guidelines for iteration one can be found in Table 4.3.

Clear goals and feedback were the elements that were given the highest score for the iteration among the elements from the Game flow model. The fact that the game followed a lot of 2D platform game standards that were familiar to players, e.g. score collectibles and quota tokens, made the goals in the game very clear. The feedback element was
Chapter 4. Results

Figure 4.5: The default motion zone layout at the end of iteration one.

<table>
<thead>
<tr>
<th>#</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Determine character and level size</td>
</tr>
<tr>
<td>1.2</td>
<td>Modify character- and game-mechanics</td>
</tr>
<tr>
<td>1.3</td>
<td>Implement a level structure that allows for several levels and a level complete state</td>
</tr>
<tr>
<td>1.4</td>
<td>Create a set of levels for testing</td>
</tr>
<tr>
<td>1.5</td>
<td>Create a health system for the player character and enemies, including a health bar</td>
</tr>
<tr>
<td>1.6</td>
<td>Implement spawn points, checkpoints and a goal</td>
</tr>
<tr>
<td>1.7</td>
<td>Make it possible to kill and get killed by existing enemies</td>
</tr>
<tr>
<td>1.8</td>
<td>Create moving platforms</td>
</tr>
<tr>
<td>1.9</td>
<td>Create bottomless pits</td>
</tr>
<tr>
<td>1.10</td>
<td>Create one type of power-ups (first aid)</td>
</tr>
<tr>
<td>1.11</td>
<td>Create a secondary game objective (quota token collectible)</td>
</tr>
<tr>
<td>1.12</td>
<td>Create an interact-able movement aid (movable crates)</td>
</tr>
<tr>
<td>1.13</td>
<td>Create further movement aids (springboard)</td>
</tr>
<tr>
<td>1.14</td>
<td>Keep track of the time played on each level</td>
</tr>
<tr>
<td>1.15</td>
<td>Update the score user interface with essential information</td>
</tr>
<tr>
<td>1.16</td>
<td>Improve the layout of the motion zones</td>
</tr>
<tr>
<td>1.17</td>
<td>Allow for altering of motion zone sizes and thresholds of motion zones in game</td>
</tr>
</tbody>
</table>

Table 4.2: Short description of the tasks created and implemented in iteration one.
Table 4.3: An overview of the evaluation regarding how well the game applied to the heuristics and guidelines of the Game flow model and the Motion guide at the end of iteration one.

<table>
<thead>
<tr>
<th>Game Flow Model</th>
<th>Motion Guidelines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
<td><strong>Grade</strong></td>
</tr>
<tr>
<td>Concentration</td>
<td>3</td>
</tr>
<tr>
<td>Challenge</td>
<td>1</td>
</tr>
<tr>
<td>Player Skills</td>
<td>1</td>
</tr>
<tr>
<td>Control</td>
<td>3</td>
</tr>
<tr>
<td>Clear Goals</td>
<td>4</td>
</tr>
<tr>
<td>Feedback</td>
<td>4</td>
</tr>
<tr>
<td>Immersion</td>
<td>2</td>
</tr>
<tr>
<td>Social Interaction</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

given a high score since feedback were given immediately on actions, both when it came to feedbacks concerning player movements and feedback concerning character actions. Score and status information were displayed in the score interface positioned in the top of the screen and feedback on progress toward the goal was provided by passing checkpoints. Regarding the concentration element, even though not many types of obstacles or other game components existed so far, the game required a high workload and a lot of stimuli were provided leading to the grade of 3 for the concentration element. A high workload was mainly required due to the motion interaction which required a lot of concentration, especially for new players. The elements regarding challenge and player skills were only given a grade of 1. The challenge of the game was not increased as the player progressed through the game and was not matched to players’ skill level. Also, no new challenges were introduced as the player traversed the levels and the number of obstacles existing in the game was low. The skill levels of players’ were improved when playing the game as interaction techniques and game mechanics were learned and mastered but no tutoring of the game were provided. The game met no of the criteria concerning social interaction, resulting in a score of 0 for the element.

Regarding the motion guidelines, the highest score possible were dealt to the consider movement’s cognitive load guideline. Few character actions existed, thus reducing the cognitive load, and immediate feedback of player movements were provided in the game in form of a shadow sprite displaying the player’s position on the screen. Support self-expression was also given a score of 5 since the character could be controlled in various ways as long as the motion zones was activated. The exploit risk guideline was not applied a score since no consideration was taken regarding physical risks in the development of this project. The map imaginative guidelines was not included in the grading either since the game interaction system did not directly perform any ‘mapping’
of the movements. The intend fatigue guideline had the low score of 1 since many of the levels created in iteration one resulted in too long game cycles and had a layout that required monotonous movements even though varying movements were preferred. The reason why the facilitate social fun guideline were given a grade of 2 while the social interaction element in the Game flow model were given a grade of 0 was because even if the game had no direct implementation related to multi-player gaming, audience could easily join the game to help the player to reach the goal of the levels in a cooperative fashion. Also, the game could easily be learned by watching someone play.

Play testing of the game after iteration one indicated that, even if many of the tasks implemented in the iteration were well suited for motion interaction, many improvements could be made. For example, the moving platforms were too hard to jump on no matter what velocity they had, especially for new players with lack of experience regarding the interaction technique and the game mechanics. Additionally, when the character was finally succeeded to be positioned on a moving platform, minimal movements would cause the character to fall down. A problem was also detected concerning the visual representation of the character. By default, the character was designed to face the in game camera when standing still and to face the running direction when moving. However, when trying to make the character run by applying motion in the corresponding motion zones, the motion zones would occasionally get deactivated causing the character to stop moving and face ‘forwards’ for a short duration before the running animation was continued. Furthermore, the levels created in the iteration were generally designed to have the start in the far left part of the level and a goal in the far right part of the level. This resulted in monotonous player movements where the player mostly had to apply motion in the right movement zone to traverse the level. It was also detected during play testing that a level design that included few paths in an upwards direction were preferable when using the motion based interaction implemented in the game since precise jumps were difficult to perform. An explanation of what is meant by ‘upward paths’ is presented in Figure 4.6. Finally, the new motion zone layout induced large improvements when it came to controlling the character; it also decreased the risk of unintentionally activating the wrong motion zone.

4.2.2 Iteration 2

Iteration two was the shortest iteration with fewest number of tasks. However, it included implementation of a procedurally generated level system improved upon in iteration three and many conclusions were drawn in the evaluation step.
4.2.2.1 Tasks

The main task in iteration two was the creation of a partly procedural level generation with the approach of generating levels by combining pre-designed level blocks. The task was introduced with the intent to increase diversity and replayability of the game and its levels. With this approach a large amount of levels could be created with little effort by combining the blocks in various ways. Much time was spent on creating a structure that generated high quality levels fulfilling Fisher’s procedural level generation requirements described in section 2.2.1.3. The generated levels also had to comply with the interaction scheme, mainly in the sense of incorporating level designs which allowed for varying body movements, see the “Intend Fatigue” guideline in table 2.5. A level design which requires the player to perform varying movements also mitigates the risk of player injuries related with chronic performance of static motions. Overall, the level structure was designed to create levels that allowed for ergonomic gestures which is an essential part of game design concerning motion based interaction systems as described in section 2.3.1.

Figure 4.7 displays the structure of the level generation. Each programmatic generated level consisted of eight blocks arranged in a layout with two rows and four columns. The continued arrow in the figure exposes the general path through the level. As demonstrated, the first part of the the level requires the player to mainly navigate the character in a horizontal rightwards direction while the second part required the player to mainly navigate the character in a horizontal leftwards direction, hence forcing the player to perform varying body movements. In addition, each block could also include paths that forced the player to perform local direction switches. The specific layout of the level was constructed to mainly contain horizontal and downwards vertical traversal through the level since play-testing from iteration one proved that traversal in an upwards horizontal direction somewhat contradicted the interaction scheme used in this game. Seven
different types of blocks were created, displayed by the dashed boxes in the figure. Each block in the level structure could be regarded as a cell and the exit and entry points between the blocks could be regarded as portals (see section 2.2.1.1 for description of cells and portals). Several blocks were created for each block category and each block type had at least one position in the level structure where it could be placed. The created blocks also had special requirements concerning the design depending on what category they belonged to. For example, blocks belonging to the start blocks category had to contain spawn-points and were placed in the green start block zone as displayed in the figure. The blocks were created in the Tiled editor and each block had a width of 25 tiles and a height of 16 tiles. The levels were generated at play time by randomizing one block from the corresponding block categories for each block position. As demonstrated in the figure, both the top-mid block zones and the bot-mid block zones could beyond their corresponding block types also randomize blocks from the top- & bot- mid blocks category.
Following list describes various requirements for blocks belonging to the different categories:

- **All blocks** - should provide a path from its entry (i.e. the boundary between the preceding block and the current) to the exit (i.e. the boundary between the current block and the next), with exception for blocks belonging to the start blocks category which need to provide paths from the spawn-point to the exit and blocks belonging to the end blocks category which need to provide paths from the entry to the goal. Further, the floor platform at the entry and exit of each block should be between two and four tiles in height to provide passable paths through the boundaries. This does not apply for top-right blocks where the exit should be in the ground and for bot-right blocks where the entry should be through the ‘roof’.

- **Blocks intended to be placed in the top row** - should not contain any bottomless pit obstacles or holes in the ground that exposes short-cuts to the bottom row. This does not apply for blocks belonging to the top-right blocks category.

- **Blocks intended to be placed in the bottom row** - may contain bottomless pit obstacles but may not contain holes in the ground that might make it possible for the character to fall out of the level.

- **Start blocks** - must contain a spawn point and should not be designed so that the player character can ‘fall off’ the level on the left side of the block.

- **Top-right blocks** - should have a two or three tiled wide hole in the ground positioned with ending at tile 24 in x-axis as an exit to the bot-right block zone. The blocks should contain a wall at the right side of the block to prevent the player character from falling out of the level.

- **Bot-right blocks** - should, in accordance with the top-right blocks, have a wall at the right side of the block. There should also be a two or three tiled wide hole in the ‘roof’ positioned with ending at tile 24 in x-axis as en entry from the top-right blocks.

- **End blocks** - should have a wall to the left of the block that prevents the character from falling out of the level and should also contain a goal component.

- **Blocks belonging to the top- & bot- mid blocks** - are special blocks that must apply to the requirements set out for both blocks intended to be placed in the top row and blocks intended to be placed in the bottom row. An additional requirement is that these blocks must be traversable both rightwards and leftwards.
Figure 4.8 presents screen captures from the Tiled editor displaying one of the top-right blocks and one of the bot-right blocks. Note that none of these blocks contains goals or spawn points and that, when applied; the level generator will place them at the top-right and bot-right block position. The portals are highlighted for demonstration purposes. The implementation of the level structure also included some altering of the in-game camera bounds settings. Without any modifications Phaser would automatically position the camera so that the character was always placed in the middle of the screen. With a level layout consisting of two rows this consequently lead to that part of the bottom row was visible when the character was located on the top row which was an undesirable behavior. The solution was to set the camera bounds, which specified what areas the in-game camera could be located within, to initially only cover the top row. When the character eventually reached the portal leading to the bottom row, the camera bounds would be modified to cover only the bottom row.

Iteration two also included implementation of a life system. The player character was initialized with a pre-determined amount of bonus lives at the start of each level and the number of bonus lives was displayed in the score user interface. Every time the character died in the level the life amount was reduced. At the point when the player died and there were no bonus lives left, instead of resetting the character to the position of the last passed checkpoint or the spawn point, the whole level was reloaded and all collected items were lost. Moreover, a bonus life power-up was introduces that increased the number of bonus lives by one when collected. A new obstacle was introduced as well, a stationary and imperishable spike obstacle that could be placed either on the floor, on the roof or on the walls.

Several minor improvements were added to the game, one of which was adding a bounce effect to the character that was triggered either when the character was damaged by an obstacle or when the character damaged any of the destroyable obstacles. Bouncing
on enemies in sequence without touching the ground is one of the most common tasks introduced in early game-play in popular platform games and is often used to increase the player’s engagement level as described in section 2.2.1.6. Since play-testing in iteration one demonstrated that the two tiles wide moving platform was hard to navigate the character to given the utilized interaction technique a new version of the moving platform was created with the width of three tiles. Other minor improvements were; displaying a ‘player died’ text when the player died and making the level complete state automatically start the next level after a specified time period. A fix were also applied to prevent the character from facing forwards when running and the active motion zone was set inactive due to lack of motion in the zone for a really short duration (see the evaluation section in iteration one, section 4.2.1.2).

Furthermore, the layout of the motion zones was altered to get a more intuitive interaction scheme with more control over the character and the applied actions. Figure 4.9 displays the layout of the motion zones at the end of iteration two, the two jump zones were merged and the new zone allowed for character jump actions on the entire top of the screen. The keys for altering the size and threshold of the motion zones implemented in iteration one were altered to apply for the new layout.

A summary of all tasks created and implemented in iteration two is found in Table 4.4.
## Tasks in Iteration 2

<table>
<thead>
<tr>
<th>#</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Create more obstacles (spikes)</td>
</tr>
<tr>
<td>2.2</td>
<td>Add a “character died” text</td>
</tr>
<tr>
<td>2.3</td>
<td>Introduce a new wider version of the moving platform</td>
</tr>
<tr>
<td>2.4</td>
<td>Make the level complete state automatically start the next level after a certain duration</td>
</tr>
<tr>
<td>2.5</td>
<td>Improve on the current motion zone layout</td>
</tr>
<tr>
<td>2.6</td>
<td>Create a procedurally generated content structure for the levels</td>
</tr>
<tr>
<td>2.7</td>
<td>Make the character bounce on obstacles when harmed or inducing damage on obstacles</td>
</tr>
<tr>
<td>2.8</td>
<td>Fix issue where player character faces forward too often when trying to make the character run</td>
</tr>
<tr>
<td>2.9</td>
<td>Implement a bonus life system for the character along with a bonus life increase power-up</td>
</tr>
</tbody>
</table>

**Table 4.4:** Short description of the tasks created and implemented in iteration two.

<table>
<thead>
<tr>
<th>Game Flow Model</th>
<th>Grade</th>
<th>Motion Guidelines</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concentration</td>
<td>3</td>
<td>Embrace Ambiguity</td>
<td>4</td>
</tr>
<tr>
<td>Challenge</td>
<td>1</td>
<td>Celebrate Movement Articulation</td>
<td>2</td>
</tr>
<tr>
<td>Player Skills</td>
<td>1</td>
<td>Consider Movement’s Cognitive Load</td>
<td>5</td>
</tr>
<tr>
<td>Control</td>
<td>4</td>
<td>Focus on the Body</td>
<td>1</td>
</tr>
<tr>
<td>Clear Goals</td>
<td>4</td>
<td>Intend Fatigue</td>
<td>4</td>
</tr>
<tr>
<td>Feedback</td>
<td>4</td>
<td>Exploit Risk</td>
<td>-</td>
</tr>
<tr>
<td>Immersion</td>
<td>3</td>
<td>Map Imaginatively</td>
<td>-</td>
</tr>
<tr>
<td>Social Interaction</td>
<td>0</td>
<td>Highlight Rhythm</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Support Self-Expression</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Facilitate Social Fun</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table 4.5:** An overview of the evaluation regarding how well the game applied to the heuristics and guidelines of the Game flow model and the Motion guide at the end of iteration two. Positive changes in the score compared to the evaluation in the previous iteration is highlighted in green.

### 4.2.2.2 Evaluation

A table displaying the gradings of the evaluated heuristics and guidelines for iteration two can be found in Table 4.5. Note that elements and guidelines highlighted in green in the table indicates positive changes in the grade compared to the evaluation in iteration one.

The improved motion zone layout implemented in iteration two made maneuvering of the character more intuitive and convenient leading to a grade increase of one for the control element in the Game flow model. For the same reason, the score of the immersion element was increased. The layout of the implemented level generator enforced more variations in player movements since the created levels required the character to navigate
in both horizontal directions, this was the reason for increasing the score regarding the intend fatigue guideline. Since all blocks created in the iteration were designed to have a rhythmic pattern, hence making each block a unique rhythm group, the score concerning the highlight rhythm guideline was increased as well.

Play testing conducted in iteration two indicated that the moving platform with increased width coped much better with the interaction scheme than the two tiles wide platform. The fix that had been applied to solve the visual inconsistencies that appeared when the character faced forward in between the running animation was successful as well. Some flaws in the game were however detected. The implemented bonus life system that forced the player to restart the level after a certain amount of deaths made the player cautious resulting in a game with lower pace and reduced player movements. Since high paced games with a game design that allow for much player movement has been proved to increase the enjoyment and engagement of games, the bonus life system only seemed to induce negative effects on the game. This also applied for the text that appeared when the character died. It was also observed that in the general case, the player would play the game while seated and interact by only moving his arms. However, motion games that results in whole-body movement interactions are proved to elicit more engagement and enjoyment than games that only requires piecemeal motion (see section 2.3.2). This fact lead to the assumption that the game in this project could be improved by altering the interaction scheme to allow for whole-body interaction. Furthermore, it was detected that status bar had no information about what level that was played and that the health bar placement and the placement of the element displaying the amount of time spent playing the level did not comply with game standards. Another observation that was made when play testing was that the levels generated could sometimes be perceived as repetitive. Sometimes the same block could appear at block positions right after each other, for example when the same block was randomized in both the top-mid block1 position and the top-mid block2 position. It also happened that the same start block was randomized in two subsequent levels, making the player believe that he had to start over on the same level he just completed. It was also decided that the monsters implemented in iteration one had to be wider as they were too hard to hit, when the monsters were small players tended to just continue through the level instead of killing the monsters. This was not preferable, since killing enemies is strongly related to player enjoyment as described in section 2.2.1.2.

Two observations were made concerning the motion interaction. The first was that video streams received from some web cameras were horizontally mirrored and had to be reversed before being used in the game. The second observation was that thresholds and required size of the motion zones highly depended on the size of the screen on which the game was displayed. On larger screens, like TV screens, the player was positioned
further away from the web camera than when playing on computer screens, making the player ‘smaller’ from the perception of the camera. As result, a motion zone layout that worked well on PC screens could be hard to use on a TV screen since the motion zones would then appear too far apart. As for the motion thresholds, values appropriate for playing the game on a PC screen would have to be reduced when playing on a TV screen since the amount of motion required to activate the zones would otherwise be too hard to produce.

4.2.3 Iteration 3

Iteration three was the last and the longest iteration and contained the largest amount of tasks. Major modifications were made to the game both structurally and appearance wise, new components were added and plenty of bugs were fixed. No altering was made to the game in this project after this iteration.

4.2.3.1 Tasks

Several modifications were made to the level generator implemented in the previous iteration and the system was expanded with various improvements. A new level layout was added that allowed the generator to create levels consisting of six blocks instead of eight. The layout had a similar setup as the previous layout, see Figure 4.7, but with only one top-mid block and one bot-mid block. The generator was adjusted to randomize between the short and the longer layout when creating new levels. The two main reasons for adding a further layout was to create variety between the levels, i.e. to prevent the levels from being perceived as repetitive, and to comply with one of the guidelines regarding motion based interactions in Table 2.5 which states that motion based games should apply short game cycles to minimize chances of fatigue. To add further variety between the levels the generator was prevented from randomizing the same block more than once within the range of two levels. For example, when a certain block was randomized in one level, it could not get randomized again in that level or in the next level. As described in the evaluation step of iteration two, section 4.2.2.2, the start blocks of the levels had a predominant impact on how repetitive the game felt. The start blocks for the levels were therefore decided to be unique; at least until all existing start blocks were used. For the same reason, a large number of new blocks were created to be used by the generator.

It is important for games to offer a degree of challenge that agrees with the skill level of the player (see section 2.2.1.4) and as stated in section 2.2.1.2, most players enjoy fast paced games with quick progression. A progression system was implemented that
increased the difficulty of the game and the level of challenge in a relatively fast pace. As demonstrated in Figure 4.10, the blocks retrieved by the level generator was structured in a specific directory tree. The blocks in each directory were named ‘blockX’, where ‘X’ was unique for every directory and started at 1. They were also ordered in increasing level of difficulty, i.e. block1 was the least challenging block and the block with the highest number was the most difficult to traverse. The level generator was modified to randomize between a certain range of blocks from each block category depending on which level that was generated. When constructing levels appearing early in the game, only easy blocks were used and for levels appearing later in the game more complex blocks were used. Two parameters were used to determine the pace of difficulty increase and the variety range. The figure shows an example where the level generator was regulated to randomize within the range of three blocks and to update the start of the range for every other level. That is, when constructing game level three, only the three easiest blocks (‘block1’-‘block3’) are used in the randomization process. The same applies when generating blocks for level four. This is indicated by the green area in the figure. When constructing level five and six, the range to randomize blocks from was changed and the generator would consequently pick blocks within the range of ‘block2’ to ‘block4’ as indicated by the red area in the figure. Note that the generator starts randomizing blocks for level three since the first two levels are pre-defined due to the in game tutorial system described later in this chapter. Two exceptions exist regarding the recently described method. Since the amount of blocks in each block category is limited, the player could reach a level that would require the generator to randomize within a range of blocks that does not exist. When that is the case, the level generator will instead be forced to randomize from all of the existing blocks in the specific category. The other exception concerns the start blocks category, blocks belonging to this category are not randomized, as already described, but are instead picked in the order of difficulty.

Iteration thee also included further improvements on the motion zone layout. It is stated that games resulting in whole body movements when interacting elicits more player engagement and are more enjoyable than motion based games that only requires so called piecemeal motion (see section 2.3.2). The aim of the motion zone layout implemented in this iteration was therefore to produce a solution that provoked whole body interactions while still managing to keep the precision of the control succeeded to be produced in previous iterations. The final result of the new motion zone layout is displayed in Figure 4.11. The zones that triggered running actions were placed at the side of the game screen similar to earlier implementations. The jump zone was however placed in the large area in between the running zones and had a much higher motion threshold than the other zones. This required the player to take a standing position and to actually jump in order reach the motion threshold required to make the character
jump. Additionally, arrows were added to the motion zones to further enhance the zones functionality. The arrows were made visible when the zones were activated. The arrow corresponding to the jump zone is visible in the figure.

Several new game components were introduced and are presented in the following list:

- Two different types of power-ups, one that made the character invulnerable when collected and one that allowed the character to perform double-jumps. The effects of both power-ups were applied over a limited amount of time. The so called invincible power-ups did not protect the character from dying when falling into bottomless pits. The power-ups were graphically designed in a way that would help the player understand their functionality. Double jump power-ups had the appearance of a shoe while the invincible power-ups were designed to look like shields. When collecting a power-up, the character would get tinted with a color corresponding to the color of the power-up for the duration that the effect applied.

- Three new types of enemy monsters. One type that had 2 health points, a velocity of 60 units and inflicted 25 hit-points damage. Another type with 3 health points, a velocity of 80 units and a damage property of 20 hit-points. The third type of monster had 3 health points, a velocity of 70 units and a damage property of 40 hit points. This type of monster was also given the functionality to throw obstacles that the player had to avoid in order to not get harmed. The monster type that existed prior this iteration had 1 health point, a velocity of 100 units and inflicted 25 hit-points damage. Additionally, all type of monsters except from the type that threw balls were made wider to be easier to jump on as a measure to the problems described in the evaluation in section 4.2.2.2. For the same reason,
a reward system was implemented that gave the character extra star scores for killing monsters. Different monsters returned a different amount of star scores depending on their difficulty.

- A trigger component in the form of a button that the character could jump on. When triggered, one or more platforms connected to the specific button would appear and open paths to otherwise unreachable platforms. Like most other components, the trigger button and corresponding platforms were added to the blocks in the Tiled editor. All buttons and belonging platforms were given unique identifier so that the game algorithm would know which platforms to enable.

- A pendulum obstacle in the form of a spear. The spear would emerge from the ground of the platform it was placed on before returning back into the ground in a repeating fashion. The player could either chose to jump over the obstacle or to try to run over it when it was under the ground. A repetition property defining the pace of the spear was set to each spear in the Tiled editor. Figure 4.12 shows an example of the spear functionality.
As suggested in both the game heuristics in Table 2.1 and in the ‘Player skills’ section of the Game Flow model, Table 2.2, a game should have tutorials or initial levels where the basics of the game are taught to the players. Iteration three included implementation of

- An unstable floor obstacle that consisted of a two tiles wide platform that would fall apart a short duration after the character stepped on it. After a few seconds the floor was reset to its original position.

- A destroyable crate, similar to the movable crate implemented in iteration one, but which could not be moved and had no gravity. The crate could not harm the player but could be destroyed when jumped at from below. When destroyed, the destroyable crate emitted a number of stars that could be collected by the character. This component was created due to the fact that jumping on coin blocks in Infinite Mario Bros were significantly correlated with the emotion of ‘fun’ according to the research described in section 2.2.1.2.

- A new platform component that reduced the drag on the character to zero. When the character entered these platforms no motion had to be applied in the right and left motion zones to keep the character moving and only jump actions needed to be performed in order to collect eventual stars or to avoid obstacles. However, the platform did not prevent the player from increasing the movement speed on the character or changing its direction if desired.
two types of tutorials, one animated tutorial that could be chosen whether to watch or not and one in game tutorial that provided explanation of the game basics while playing the game. The animated tutorial described how to use the interaction technique while at the same time displaying an animation of how to perform the described actions and how the character would move when the actions were applied. The in game tutorial was applied in the two first levels of the game. Unlike the other levels which were procedurally generated, the first two levels were decided to be pre-defined and to utilize the short level layout (two rows and three columns). They were composed so that the player could easily traverse the level and reach the goal. The first level was designed to only require the character to move horizontally, i.e. to run, and an animated stick figure displayed how the running action should be performed (see Figure 4.13). A glowing effect was also applied on the corresponding motion zone in which the player should interact. The second level introduced jump actions and included a system that made sure that the player apprehended the jump technique before being able to reach the goal. This system was constructed by including two destroyable crates that had to be destroyed, by jumping on the from below, before a wall platform disappeared and opened up a path to the goal (see Figure 4.14).

A new character action was implemented, a ground stomp attack that was triggered when the player reached a high enough threshold in the jump zone, this threshold was considerably higher that the threshold for performing normal jumps. A ground stomp animation was added to the character in mid-air as a feedback indicating that the stomp action was about to be performed. When the character landed on the ground, earth and stones were thrown up from the ground in a narrow range around the character killing all enemies within the range. The ground stomp action was mainly introduced since platform games with a high number of character actions increases the level of freedom. 

Figure 4.13: The figures shows two frames from the in game tutorial, level one, where the player is tutored how to make the character run.
Chapter 4. Results

55

(a) Jump sequence one. (b) Jump sequence two.

Figure 4.14: The figure shows two frames from the in-game tutorial, level two, where the player is tutored how to make the character jump. The wall on the right side of the character is opened when both crates have been destroyed.

experienced by the players compared to platform games that only offers few character actions (see section 2.2.1.5).

A reward system for the quota tokens was established. The total amount of quota tokens in each level was displayed in the score user interface. If the complete set of tokens in a level was collected, the next coming level would get randomized by the level generator as usual, but with different graphics indicating that the level was a bonus level. Every star collected in a bonus level was multiplied by 2. A screen shot from a bonus level is found in Figure 4.15.

Improvements were made to the graphics and various audio elements were added to the game. Two different image backgrounds were added along with a parallax scrolling tile sprite illustrating distant mountains. Several tweens and emitters were configured to give feedback on important events, e.g. displaying fireworks when the character finished a level. The shadow sprite was altered to appear in front of the platforms, as in earlier implementations, but behind the character and behind background and foreground components such as trees or flowers. A pixelate filter was also added to the shadow sprite for aesthetical purposes. The text that was implemented in iteration two and appeared when the character died was removed due to its negative effects (see section 4.2.2.2). Furthermore, an element was added to the score user interface displaying which level that was played and the components of the score user interface were rearranged to be more consistent with game standards by switching placement on the health bar and the level time. A death count was also implemented and used for providing information in the level complete state about how many times the character died in the played level. Regarding audio elements, music was created and applied to the game according to the suggestions in section 2.2.4 and audio effects were incorporated. A general rule regarding audio elements is that games should offer an option for adjusting volume on
music and audio effects independently; hence a pause menu was introduced where such functionality existed. It also included a mirror setting that would mirror the image perceived by the web camera in case it was horizontally reversed. Additionally, a setting was added in the pause menu that allowed the player to choose whether the game was played on a desktop display or on a television screen. Switching between PC and TV mode resulted in adjustments of motion zone sizes and thresholds. The reason for adding this functionality is explained in the evaluation of iteration two, section 4.2.2.2. The pause menu is shown in Figure 4.16.

Further fixes and improvements were added to the game in the iteration. A third-party image sharing software created in another project was incorporated in the game. Screen captures of the game were automatically taken at gameplay and displayed by the level complete state. The players could then choose from a set of images and click on a share button that allowed the players to share the moment on Facebook. Figure 4.17 features the game screen displayed by the level complete state after implementation of the image sharing functionality. The life system implemented in iteration two was removed completely from the game due to reasons explained in section 4.2.2.2. A timer
Chapter 4. *Results*

![Screen shot of the pause menu.](image)

**Figure 4.16:** A screen shot of the pause menu. The volume of audio effects (SFX) and music could be adjusted independently. There were also options for changing or muting the master volume. The option for mirroring images from the web camera was provided at the top of the pause menu and changing default values for the motion zones were made in the bottom of menu.

was added to the game that restarted the game at the first tutorial level if no motion was detected over a period of approximately one minute. The restart procedure could be toggled on and off and was implemented for situations where the game was made available on big screens in public environments where new players would consistently try out the game. In addition, a trail effect was implemented on the shadow sprite from the images retrieved from the web camera to further minimize the visual inconsistencies produced when a motion zone was disabled for a really short duration before being activated yet again. The trail effect functioned by adding the motion image from a number of previous frames to the current shadow sprite.

A summary of all tasks created and implemented in iteration three is found in Table 4.6.
# Tasks in Iteration 3

<table>
<thead>
<tr>
<th>#</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Improve the graphics in the game add add audio elements</td>
</tr>
<tr>
<td>3.2</td>
<td>Add tweens and emitters to game events and actions as a form of feedback</td>
</tr>
<tr>
<td>3.3</td>
<td>Update the graphical score user interface with a component indicating what level is played and rearrange the order of the score elements to correspond with game standards</td>
</tr>
<tr>
<td>3.4</td>
<td>Display the total amount of quota tokens in the levels in the score user interface and create a bonus system for collecting the complete set of quota tokens</td>
</tr>
<tr>
<td>3.5</td>
<td>Remove the ’player died’ text</td>
</tr>
<tr>
<td>3.6</td>
<td>Yield rewards for killing monsters</td>
</tr>
<tr>
<td>3.7</td>
<td>Create a pendulum obstacle (spear pendulum)</td>
</tr>
<tr>
<td>3.8</td>
<td>Implement unstable floors</td>
</tr>
<tr>
<td>3.9</td>
<td>Implement a destroyable crate that yield stars when destroyed</td>
</tr>
<tr>
<td>3.10</td>
<td>Implement a platform with no ’friction’</td>
</tr>
<tr>
<td>3.11</td>
<td>Introduce two new types of power-ups (double jump power-up &amp; invincibility power-up)</td>
</tr>
<tr>
<td>3.12</td>
<td>Implement a trigger component that activates invisible platforms</td>
</tr>
<tr>
<td>3.13</td>
<td>Introduce various kinds of new monster enemies</td>
</tr>
<tr>
<td>3.14</td>
<td>Make the enemies wider to be easier to hit</td>
</tr>
<tr>
<td>3.15</td>
<td>Update the layout of the motion zones to allow for whole body interaction</td>
</tr>
<tr>
<td>3.16</td>
<td>Add tutoring to the game</td>
</tr>
<tr>
<td>3.17</td>
<td>Fix the problem with repetitive levels</td>
</tr>
<tr>
<td>3.18</td>
<td>Introduce a new level layout (3x2 blocks)</td>
</tr>
<tr>
<td>3.19</td>
<td>Implement a system for game progression</td>
</tr>
<tr>
<td>3.20</td>
<td>Introduce a new character action</td>
</tr>
<tr>
<td>3.21</td>
<td>Incorporate game image sharing functionality</td>
</tr>
<tr>
<td>3.22</td>
<td>Design several level blocks to be used by the level generator</td>
</tr>
<tr>
<td>3.23</td>
<td>Make the shadow sprite appear behind background and foreground objects also behind the player character</td>
</tr>
<tr>
<td>3.24</td>
<td>Improve the interaction scheme by implementing a ’trail’ effect on the shadow sprite</td>
</tr>
<tr>
<td>3.25</td>
<td>Add a pixelate filter to the shadow sprite, for aesthetical purposes</td>
</tr>
<tr>
<td>3.26</td>
<td>Implement a pause menu containing setting for mirroring, audio volume adjustments and motion zone layout altering</td>
</tr>
<tr>
<td>3.27</td>
<td>Implement a death counter and display number of times the character died before completing the levels</td>
</tr>
<tr>
<td>3.28</td>
<td>Add an option that restarts the game if no motion is detected for a certain period of time</td>
</tr>
<tr>
<td>3.29</td>
<td>Remove the life system implemented in previous iteration</td>
</tr>
</tbody>
</table>

Table 4.6: Short description of the tasks created and implemented in iteration three.
Chapter 4. Results

59

(a) Before choosing an image.

(b) After choosing an image.

Figure 4.17: The figures shows what was displayed by the level complete state after implementation of the image sharing functionality.

<table>
<thead>
<tr>
<th>Game Flow Model</th>
<th>Motion Guidelines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Grade</td>
</tr>
<tr>
<td>Concentration</td>
<td>5</td>
</tr>
<tr>
<td>Challenge</td>
<td>4</td>
</tr>
<tr>
<td>Player Skills</td>
<td>4</td>
</tr>
<tr>
<td>Control</td>
<td>5</td>
</tr>
<tr>
<td>Clear Goals</td>
<td>4</td>
</tr>
<tr>
<td>Feedback</td>
<td>4</td>
</tr>
<tr>
<td>Immersion</td>
<td>5</td>
</tr>
<tr>
<td>Social Interaction</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.7: An overview of the evaluation regarding how well the game applied to the heuristics and guidelines of the Game flow model and the Motion guidelines at the end of iteration three. Positive changes in the score compared to the evaluation in the previous iteration is highlighted in green.

4.2.3.2 Evaluation

A table displaying the gradings of the evaluated heuristics and guidelines for iteration three can be found in Table 4.7. Note that elements and guidelines highlighted in green in the table indicates positive changes in the grade compared to the evaluation in iteration two.

Almost all of the elements from the Game flow model were given increased scores at the end of iteration three. The concentration element acquired 5 points due to all the stimuli provided in the game in form of varying levels, varying graphics and sound effects and a large amount of different game components. The progression system implemented in iteration three raised the score of the challenge element to 4 and the player skills element was increased due to the new tutorials. Also the feeling of control of the game were
increased as a result of the new interaction system and the increased width of monsters. The image sharing functionality raised the social interaction element to a grade of 1. Regarding the motion guidelines, embrace ambiguity retrieved a score of 5 since all of the criteria concerning the guideline were met. Facilitate social fun got increased to 3 points since the new interaction system allowed for whole body movements.

Play testing were carried out in iteration three as well and mostly positive observations were made. The implemented ground stomp action worked really well, it encouraged the players to perform more movements and consequently also acquire more exercise. At the same time, the ground stomp could not be performed constantly since the player would get too exhausted. The new interaction system did allow for more movement too, it was also more intuitive than the motion zone layouts from the earlier implementations. Concerning the progression system, the optimal setting to prevent the game from seeming repetitive would be to have plenty of blocks for each category, a large randomization range and to increase the randomization range for each level. The reason why this was not the case for this project was due to the restricted amount of blocks that could be created within the time frame of the project.
Chapter 5

Discussion

This chapter discusses the result of this project and the methods that were used to acquire the result. The result section provides an analysis of the final outcome of the game, implemented structures and game components that coped well with the applied motion interaction scheme are emphasised and compared to those that were not as successful. Discussion about possible improvements and future work is presented as well.

5.1 Results

Due to the layout of this project, much discussion and arguing about specific design decisions and conclusions were carried out in the evaluation phase of each sprint (see section 4.2.1.2, section 4.2.2.2 & section 4.2.3.2). This section will therefore mainly contain discussion about the final result of the game rather than discussion about specific design decisions. However, some particularly significant specific decisions will be analyzed as well.

The majority of tasks implemented during the project had positive impacts on the game. The game was analyzed using motion guidelines (see Table 2.4, Table 2.5 & Table 2.6) and game heuristics (see Table 2.2) and it was confirmed that the game increasingly agreed with guidelines and heuristics as new tasks were introduced and implemented. Positive and negative impacts on the game were also detected during play testing. The fact that the majority of features from regular platform games functioned well in the game developed in this project indicates that with various minor arrangements, entertaining motion based platform games can be created.
A problem with motion based interaction in games is that the precision of the control is generally not as good as with standard desktop computer controls. This was the main reason for the previously mentioned measure of increasing the size of certain game components. The precision is an important aspect to keep in mind when designing action based platform games, since these games often uses precision to incorporate challenge (see section 2.2.1.4). Platform games relying on motion based interaction must therefore primarily use other means when incorporating challenge. One good way of increasing challenge in motion based platform games would be to introduce more complex rhythm patterns, for example, the player have to perform timed jumps but the precision of the landing spot is not that important. Another way is to make long patterns as suggested in section 2.2.1.4. This would also increase the challenge due to player fatigue which falls under the Intend Fatigue guideline in Table 2.5. However, it is important to avoid fatigue in the game when not used as a way to increase challenge.

Most of the introduced game components were extracted game components frequently used in popular 2D platform games and, most of the time, only minor modifications were needed to make the components cope with the motion interaction used in this project. For example, the original implemented size of enemy monsters and moving platforms that would have coped with non-motion based platform games were expanded in width in iteration two and iteration three. This made it easier to jump onto the moving platforms or to kill the monsters and was necessary for reducing the risk of player frustration as well as for maintaining a high game pace. When game components, such as obstacles, induce challenge higher than the current skill level of the player, the player tends to control the character with more caution and spend more time standing still. According to the information in Table 2.3, time spent standing still with the character is a ‘feature’ strongly related to frustration. Furthermore, a cautious player will perform less movements, as described in section 2.3.1 this may or may not influence the enjoyment of the game but it contradicts with the aim of incorporating exercise in the game. These factors make a well-balanced progression pace especially important for motion based platform games.

As described in section 4.2.3.1, the progression in the developed game consists of generating levels from blocks with increasing difficulty. The more levels the player completes the harder the generated levels will be. This is a system that works quite well for the specific procedurally generated level structure, but further improvements can be made regarding the progression. One improvement would be to alter various parameters for obstacles and other game components as the player progress through the game and reach higher levels. For example, by altering the amount of damage enemy monsters or other obstacles inflict or increasing the velocity of the moving platforms. Additionally, the size of components could be decreased in higher levels to improve the progression, but
in that case a considerable amount of testing has to be performed to make sure that it doesn’t affect the game in a negative fashion due to the lack of precision in motion based interaction.

The motion zone layout was altered several times during the project with the intention to improve on the interaction system. The motion zone layout introduced in iteration two provided many benefits over the earlier layouts, the character could be controlled with quite good precision and earlier problems where players unintentionally activated wrong zones were mitigated. In iteration three major modifications were made on the layouts and settings of the motion zones. The new system allowed for whole-body interaction and henceforth resulted in positive effects for the enjoyment of the game and the engagement level of the players (see section 2.3.1). Presumably, the new layout could also induce more health benefits than previous versions according to research in how active video games promote physical activity in children and youth [31]. Of course, previous motion zone layouts could be played in a standing position but then primarily only included movement of the upper body. The final layout more or less enforced the player to perform whole body activities; this resulted in movements that were more enjoying to perform and more entertaining to watch at the expense of faster player fatigue. Additionally, the interaction was more intuitive since it enforced a higher degree of kinesthetic mimicry (see section 2.3.2. One possible drawback was that the new layout enforced the player to take a standing position since the control was very restricted while seated. A possible solution would be to allow the player to choose whether to player seated or not and apply the corresponding motion zone layout thereafter.

The ground stomp character ability proved to be a successful addition to the game. While popular platform games tend to provide a large number of character actions over relatively few number of buttons thereby increasing the perceived level of freedom in the game [9], motion based games must consider movement’s cognitive load (see Table 2.4). Multiple actions require an increased amount of cognitive attention, which is a limited resource. Movement based platform games should therefore provide a limited amount of character actions, or at least introduce the actions in a pace that let the player learn the basic actions before new ones are being introduced. Since the ground stomp ability is basically just an extension of the jump action which is introduced at an early stage in the game, the cognitive load is not overdrawn. It is also stated that platform games should have no interruption in between character actions (see section 2.2.1.5). In non-movement based games ground stomp abilities often comes with a short delay before the character is able to continue moving or perform other actions to prevent the ability from being overly abused. However, the ground stomp ability implemented in the game developed in this project have no assigned interruption, instead the natural concept of fatigue is considered a restriction for how often the action is performed. That
is, since the player has to enforce a high amount of motion in the jumps to activate a
ground stomp ability he will optimally spare energy until the action is needed in order
to proceed in the level.

One feature commonly used in standard platform games that did not fit into the design
of motion based platform games was the bonus life system. The system had the effect
of making the player cautious resulting in a game with lower pace and reduced player
movements. Since high paced games with a game design that allow for much player
movement has been proved to increase the enjoyment and engagement of games, the
bonus life system only seemed to induce negative effects on the game. Bonus lives were
therefore decided to be removed and the player was given an unlimited number of retries
on each level upon death.

Two different types of tutorials were implemented in the game, one animated tutorial
and one interactive in-game tutorial that were incorporated in the first two levels of the
game. The in-game tutorial that allowed the player to learn by trying was detected to
provide a clearer explanation of the interaction system and the game mechanics. The
animated tutorial could work as a complement when tutoring the players but the in
game tutorial was the method of choice in providing a comprehensive and accessible
game experience for new players.

According to Table 4.7, the game fulfilled many of the usability principles regarding
both games in general and motion based games. The ‘social interaction’ element and
the ‘facilitate social fun’ guideline had a score of one and three respectively. The game
is easy to learn by observing and the movements required to be performed by the player
are fun watching. The image sharing functionality also contributed to the score of the
design principles. What is missing in the area of social interaction is a multiplayer
option. Of course, bystanders can join in on the game to cooperate with the player to
finish the levels but to reach a higher score in both categories the game would have to
offer a multiplayer option, either to by playing against each other on line or to provide
an alternative interaction layout that allowed for multiple players to play together using
the same web camera. Another improvement of the game would be to implement more
of the strategies described in the ‘celebrate movement articulation’ guideline. This could
be by adding visual feedback when a player performs certain movements. For example,
applying more movement that usual in a motion zone could result in a graphical feedback
where the character gets blurry due to its speed. The ground stomp ability might meet
some criteria of the guideline but as described in the guideline, celebration of movement
articulation does not have to affect the game mechanics.
5.2 Method

Overall, the method applied in this project functioned well when it came to establishing concepts and components that coped with motions based platform games by creating an actual game. Even though no statistical calculations were made to obtain answers to the research question, plenty of suggestions and heuristics regarding the design of motion based platform games could be elicited.

One main drawback with the approach was that much of the assumptions that were made were based on my own opinion and my interpretation of the literature. By introducing a test group and by performing recurring group usability tests along the development process of the product the results would probably have been even more reliable. However, it should be added that advisors with expertise in the area of games and motion interaction were involved in many of the significant decisions and that informal usability tests with smaller test groups were conducted during the project. Additionally, more usability tests in general would have been preferable to confirm the validity of the results.

Also, if more time had been devoted on the project, additional iterations could have been executed to perfect the game. For each iteration cycle, the evaluation of the implemented features and comparison to literature and heuristics resulted in new findings. Thus, with more cycles, further findings would probably have been made as well as affirmation of the already established conclusions leading to higher validity of the acquired results.

If I had the chance to redo the study, I would have spent more time searching for and performing self-studies on existing motion based platform games available on platforms such as Sony EyeToy or Microsoft Kinect. Even though the interaction system is not exactly the same, commonly used design strategies as well as flaws in the design could probably have been elicited and used as guidelines in the creation of the product implemented in this project. Another good idea would have been to identify problems in game reviews as done in the article Heuristic Evaluation for Games: Usability Principles for Video Game Design [32]. As with the self-studies, the identified problems could have been regarded in the development of the game.

Regarding the reliability, if the study was repeated using the same method utilized in this project, the results would probably have been similar. Many of the conclusions would have been the same but since the focus area is quite large, it is not improbable that further findings would have been made that is not addressed in this project.

The literature that constituted the foundation of the study was extracted from both industrial and scientific sources. All sources were thoroughly examined before being applied in the study, especially the industrial literature. Mostly well-known industrial
literature cited by several other articles were used in this study, this way confirmation of the validity of the source could be made and eventual criticism of the source could be found in the citing articles. Arguments for the conclusions provided in industrial literature cited by few other articles was examined and if the results was not based on facts, the content was considered as hypothesis.

5.3 The Work in a Wider Context

In this study, a whole-body motion based game accessible through the web browser was developed. As opposed to general games, games that require body motion for interacting has been proven to come with many physical health benefits, especially for games that involves whole-body interaction. In an article by Biddiss & Irwin, it is discussed that many children does not meet the recommended level of physical activity and how physical inactivity is a risk factor for several chronic health conditions, such as diabetes, cardiovascular disease, and cancer. By replacing regular games with active video games the amount of physical activity for children could be increased [31]. Research also indicates that motion based video games could have positive effects in the rehabilitation of cancer by promoting physical activity [33]. Full-body motion based games may also involve mental health benefits. For example, it is established that motion based games have a positive effect on mood and emotional well-being of older adults [34].

A risk with video games in general is that some players perceive certain games to have a simulated meaningfulness and might spend much time playing a game they think is important instead of performing more critical matters, like applying for jobs or studying for tests in school. The risk of video game addiction still pertains for motion based games, but player fatigue may in this case introduce a natural barrier for how much time is spent playing.
Chapter 6

Conclusion

This thesis presents a case study with the aim to improve on an existing motion based platform game project and to elicit platform game features coping with the utilized interaction scheme. An extensive qualitative literature based study was conducted covering literature in the area of game design, platform game design and motion based games to extract guidelines and to suggestions needed for creating a proof of concept platform game with motion based interaction. The game was evaluated and tested in rounds to achieve a high entertainment value and to find features and designs applicable for the game genre.

The game developed in this project is a good guideline for how action-based platform games with motion interaction should be designed. Adhering the suggestions presented in the study when developing a motion based platform game will increase the entertainment value of the game and most likely ensure player satisfaction. The interaction scheme incorporates whole-body interaction and immediate feedback is prioritized over natural interaction.

The created platform game features a procedural generated level structure with a progression system suited for motion based platform games. The game also includes an in game tutorial and several game components adapted for motion based platform games. During development and evaluation of the game, the majority of features and components utilized in non-movement based platform games have proven to be applicable, with minor modifications, for motion based platform games as well. Only few features available in standard platform games were found to contradict the design of motion based platform games.
Bibliography


På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© [Tim Ziegenbein]