Bachelor's thesis

Degree programme in Information Technology

Information Technology

2016

Arilesere Sharaphadeen Olakunle

3D LEVEL DESIGN USING UNREAL DEVELOPMENT KIT (UDK)

– The Mudskipper Family as case study.

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Degree programme | Information Technology

June 2016| 48

Instructor: Granholm Patric

Arilesere Sharaphadeen Olakunle

3D LEVEL DESIGN USING UDK

The gaming industry has over the years enjoyed a rapid growth cutting across various demography. This growth has brought about the need for easy creation of games, the result, game engines with predefined assets and features that ensure ease of development. Amongst the numerous phases of development or design a game is the “Level Design”.

Level design is all about the creation of levels and gameplay either through the use of geometries, level editors or AI scripting and placement, all of which requires artistic as well as architectural skills.

This thesis examines level design from three perspectives, the first of which delves into the history and theory of level design as well as the role played by level designers and how much of an effect it has in the overall production of a game.

The second deals with the requirements of designing levels using the Unreal Development Engine - a toolkit found to be useful for developers all through the production phase of a game. The final aspect of this thesis makes a case study of a project done using the Unreal Tool. The project, an adventure game titled “The Mudskipper Family” designed for the 4+ demography.

An iterative development approach was applied during the course the project and hopefully this work would be of benefit to those who intend to know more about level design and what it entails.

KEYWORDS:

Level Design, Unreal Development Engine, the Mudskipper Family.

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ
TURUN AMMATTIKORKEAKOULU

Degree programme in Information and communication Technology | Level Design

June 2016| 48

Instructor: Granholm Patric

Click here to enter text.

FORWARD

My profound gratitude to God for having seen me through the entire degree experience. My thanks goes to good number of individuals who helped make this thesis a reality either by way of supervision or lectures. Individuals like Jeff Cook, Kees Hogenhout, Marcel Verheij, all of whom exposed me to understanding game concepts, design and creation as I know today. Indeed the knowledge they imparted proved invaluable to the successful completion of the project.

There would be little or no justice done should I fail to appreciate the efforts of my supervisor Granholm Patric, whom amidst extremely busy schedules, found time to guide me all through the thesis, and my folks – Mr and Mrs Arilesere – whose words of wisdom not only encouraged me towards actualizing my goals but also uplifted me when I hit rock buttom.

2016 Turku

Sharaphadeen Olakunle Arilesere

ASIASANAT:

TABLE OF CONTENT

PICTURES

TABLES

LIST OF ABBREVIATIONS (OR) SYMBOLS

UDK	Unreal Development Kit.
UDE	Unreal Development Engine

1 INTRODUCTION

In the gaming industry, the position of a level designer is hardly recognized as it is seen as a void to be filled either by a 3D artist, an environment artist or a game designer. There is the assumption that a level designer does not bring anything proprietary to the completion of a game, the same as was said of the position of a user experience in many other industries.

With the current growth rate of the said industry, it becomes increasingly difficult to meet consumer demands at due dates while ensuring the product meets expectation. Even with this difficulty, most companies remain oblivious of an obvious solution which lies in the versatility a level designer brings to the fore.

This thesis would hopefully help readers understand what makes the postion of a level design crucial and why it must be occupied by none other than a level designer.

It begins with a brief history of level design, its work flow and how they have evolved overtime. It goes further to explain the roles of a level designer in any project, how this role may sometimes defer based on the nature of the said project and how it is often misconstrued with other partly related roles e.g game designer.

In addition to the above, this piece examines the procedural structure of designing a level using literatures and professional write-ups from successful gaming companies and/or veterans in the gaming industry. It then analyzes the game “The Mudskipper Family”, the levels therein and the implementation of its final downloadable version using Unreal Engine 3. As a level designer in the game, technical examples of challenges faced during design (from abstract to concrete) and detail the approaches undertaken to have them resolved would be cited.

In accordance of same, there is a further elaborate on as many components of level design as possible with more emphasis on how level design can make or mar the overview of a game.

2 LEVEL DESIGN THEORY

In a bid to understand this piece, it becomes imperative to expatiate on often repeated terminoligies used in the course of level design.

2.1 Terminologies

"Level design is at times a highly specialized and often technical field. This naturally leads to a certain amount of jargon and field-specific terminology"

Kremer Rudolf (2009)

- **AI placement** – This can be seen as adding detail to the world, like the use of props, and it can be seen as something that directly adds to or influences gameplay [1].

- **Character design document** – It is specifically intended to record the design of one character who appears in a game, most often an avatar. Its primary purpose is to show the character's appearance and above all her move set—a list of animations that documents how she moves, both voluntarily and involuntarily [3].

- **Colliders** – Colliders are a component of objects in game engines that simulate the interaction between physical objects [6].

- **Game** – This is an often predetermined, agreed-upon set of rules, which are designed to facilitate gameplay. The motivation behind the creation of a game itself can be diverse, for example including commercial, educational, artistic, or other elements [1].

- **Game design** – Game design consists of four steps: Imagining a game, defining the way the game works, as a system of rules, describing the elements that make it up and transmitting this information to the other members of the team [2].

- **Height Map** – This is a grayscale image – found in both game engines and 3D modelling applications – whose pixel values are mapped to the height of corresponding vertices of the terrain geometry.

- **High concept document** – This type of document puts key ideas down on paper in a bite-size chunk that can be read in a few minutes. Like a résumé, it should be short—not more than two to four pages long [3].

- **Level** – A level ordinarily refers to a portion of a video game, usually with its own victory condition, that the player must complete before moving on to the next portion. Levels are often, but not always, completed in a prescribed sequence. In storytelling terms, levels may be thought of as chapters; in war games, they are missions; in fighting games, they are individual bouts; in simulations, they are scenarios. Used with a qualifier, however, the word may take on a different meaning. See character level [3].

- **Level design** – This can be said to be the process of constructing the experience that the game offers directly to the player, using the components provided by the game design: the characters, challenges, actions, game world, core mechanics, and storyline if there is one [3].

- **Level design document** – This document allows one to structure a level well in advance of level creation and gives one a chance to prototype on paper [1].

- **Map** – A map is basically a level. Depending on the scope of a game project, the number of maps a game has will vary [4].

- **Package** – This is a collection of actors, which is what UDK calls pretty much anything one can put into a level [4].

- **Pathfinding** – Pathfinding is an artificial intelligence technique for finding the most efficient route from one point in a landscape to another while avoiding obstacles along the way [3].

- **Playtesting** – It refers to evaluating a level by having people play it many times to test for experiential and technical functionality [6].

- **Scripting** – This is the interface through which the designer can actually change how stuff works in the game, determine new behaviors, or handle special interactions, such as triggers, locks, and anything else one can imagine (within the scope of what was exposed by the engine creator) [5].

- **Storyboard** – A storyboard is a sequence of still images that show what will happen in a proposed animated sequence, such as a movie or cartoon. In game design, storyboards are used to show what will happen in a cut scene [5].

- **Tessellation** – With respect to terrains, tessellation refers to the distribution of polygons faces across the surface of a terrain which is influenced by the proximity of the player to such surface.

2.2 Overview Of Level Design History

"Now it is the beginning of a fantastic story!!

Let's make a journey to the cave of monsters! Good luck!"

Opening Screen, Bubble Bobble (1986)

The history of games is as old as that of origin of man and its core – i.e. interactivity – has always been an integral part of cultures and traditions upon which rules sets were designed. Board games were amongst the earliest of games which evolved over centuries both in rules and designs owing to the variation in cultures adopting them. Although ancient board games were primitive in nature, they still had a good element of detail as well as gameplay. Early games such as the *Royal Game of Ur and Senet*, both created between 3500 and 2500 bce, were race games whose boards reflected the pattern that players had to traverse to win the game [6].

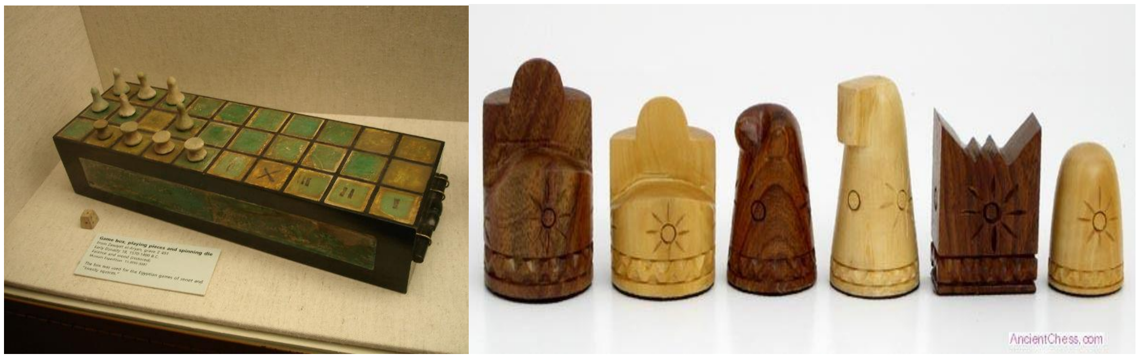


Figure 1. Pictures of the earliest senet board game and ancient chess pieces (www.windowintothebible.com).

An equally old board game which dates back to the 12th century and is still being played, is *chess* or what was referred to then as *shatranj.* Chess features a specific ruleset on what each piece may do [6], thereby emulating and triggering decisions similar to what would be made while in a battle. These ruleset were documented by Al-Sali, a legendary *shatranj* player, in what became the first scientific book – titled *Kitab Ash-Shatranj (Book of Chess)* – ever written on chess strategy.

In more recent times, games such as backgammon and other similar board games have been modelled after *Senet,* albeit with totally different set of rules – varying in complexity – and better enhanced elements. While there still exist some similarities between these ancient and modern games, the original rules have (unfortunately) been lost owing to a lack of documentation.

The importance of retracing the history of game design in relation with that of level design lies in understanding the rules and gameplay parameters, hence the argument “how else can we construct a level if we do not already know what rules it has to facilitate?” [1].

While there are those who opine that, the first examples of “playfield design” started back in the days when pinball was becoming a national pastime [7], others maintain that “level design is applied game design” [1] and because ancient board games had game designs, it serves as proof that level design is just as old as the very first game ever.

In summary, the aforementioned historical examples shows that level design is not exclusive to video games, rather, can be found in any/every game and works in unison with game designs.

2.3 Goals of Level Design (WHY?)

"Experience is key when creating level design ideas"

Dario Casali (2008)

It is imperative to discuss the goals of level design owing to the degree of codependency that exists between it and game design. This ambiguity needs to be cleared in the hope that the activities of a level designer be clearly spelt out.

The primary goal of level design lie in the augmentation of space with information. One of the ways of feeding these information to players during play is by establishing rewarded patterns of game spaces (or modular assets) which continually motivates players to further explore the game for more rewards. This goal can be categorized into two, namely; external and internal goals.

2.3.1 External Level Design Goals

These are all kinds of goals taken into consideration to facilitate the success of a game. Some are independent from direct gameplay considerations, in-so-far as they refer to goals outside of the gameplay experience [1].

The external goals include:

- It must appeal to both genders
- It must stay true to the brand image
- It must push new engine technology
- It must have a rating higher than 75%

Although these (external) goals do not determine the degree of fun/interactivity in gameplay, they serve as the foundation upon which any commercial game is built sequel to production.

2.3.2 Internal Level Design Goals

Internal level design goals are derived from direct gameplay hence, contributing to the overall experience of the game. They incude:

- Teaching the player how to have fun with game
- Giving the player a sense of achievement
- Empowering the player
- Granting reward for exploration
- Maintaining player's suspension of disbelief.
- Avoid breaking the player's suspension of disbelief
- Provide addictive elements into the gameplay

The above goals, if properly followed, defines how good a level is. It serves as a benchmark which a level designer can continually return to in a bid to ascertain the degree of completion/success of the goals.

2.4 Roles Of A Level Designer (WHAT/HOW?)

"The level designer creates the gameworld and engineers the player's gameplay experience,working in the service of an overall vision that is typically crafted by the Game Designer"

Ernest Adams (2003)

Break into the game industry: How to get a job making videogames.

Although the goals of level design had been explained prior, the *roles of a level designer* somewhat defers, not only because they define what/how a level designer goes about his work, but also because these roles are often based on the perspective from which it is viewed. These roles could be genre specific – i.e the role of a level designer in a game like ***Guitar Hero*** varies from that of a game like ***World of Tank*** – or unversal and they elaborate more on the key features mentioned in the definition of a level designer. A level designer creates the following essential parts

- *Creating the initial conditions of the level:* A level designer determines the amount of resources available to the player (within the game space) as well as their hidden location on the map

- *Setting challenges within the level:* The sequence of a challenge, also referred to as level design structure is defined by a level designer and can be approached in a number of ways within the level. This could be semi-linear, linear or non-linear and would be further explained in chapter 2.5

- *Creating the termination conditions of the level:* This is characterized in terms of victory or loss and therefore varies from one game to another.

- *Determining the aesthetics and mood of the level:* Aesthetics and mood of a level is a way through which a level designer not only speak to the sub consciousness of a player but immerse the player into the game. This can best be understood by examining the aesthetics of an actual game. Ubisoft's Tenchu: Shadow Assassin has a perfect quintessence of good aesthetic; through the use of shadow space, low-lights and sound, the game captures the essence of being a ninja thereby allowing players to be sensitive to happenings within the game space.

  The same hold true for constructing a level in a survival horror quest such Resident Evil. The experience a player gets playing such a game with features like poor lighting, eerie sounds, disarrayed environ, wreckages and a sense of isolation, would be totally different with those features absent. We can learn many things from aesthetic principles in regard to making a visual scene pleasing to look at, but the deeper lesson lies in the fact that aesthetics gives us a measure of control, a tool, for influencing the audience's mindset [1].

- *The space in which the game takes place:* If the game includes a simulated space, as most do, then level design includes creating that space using a 2D or 3D modeling tool. While game designers determine what type of asset will be in the game world, level designers determine precisely what features will be in each level of the game world and where these features will be. Level designers take the game designer's general plans for levels and make them specific and concrete [3].

2.5 Level Design Structure and Methodology

"If you cannot measure it, you cannot improve it."

Sir William Thomson (Lord Kelvin)

A level's structure and methodology are important early choices for a level designer, and they can have a large impact on the actual creation of the levels. [1]. This is the next step upon deciding on the content required in a level and their placement within the game heirarchy. There are three level design structures namely; semi-linear, linear and non-linear and are explained below in their respective orders.

Semi Linear Structure: Most free roaming games with an outdoor environment or a large map fit for exploration often have semi-linear structure (such as triggered event within the game which gives players the choice of selection from a list of options resulting to an alternative story line*)* and therefore not confined to any specific order for a certain period of time.

The advantage in this sequence lies in the fact that it gives the level designer a reasonable amount of directorial control over events the player would experience but leaves enough room for make-beliefs, with the player thinking he is in control

– by giving him the *illusion of choice*. Its downside however, is an over-the-top expectation of freedom by the player, a situation which often leads to disappointment.

Linear Structure: Linear challenges on the other hand, follows a strictly laid out sequence of event which do not support any form of deviation whatsoever. Progression is hinged on completion of certain gameplay in the predefined orders.

Although the major advantage of linear challenge is the level designer's total directorial control (which detremines pacing , consistency, story development, learning curve etc), its disadvantage lies in the exact same reason but from the players perspective. A game that makes players feel constrained, if not cleverly done, would result to boredom and resentment.

One of the many ways a level designer shifts the attention of a player away from these constraint is through *pacing and play pyscology.* For example, **TETRIS** only present the next puzzle piece after the previous piece is utilized and over time, these pieces are presented rapidly thereby stimulating the player to react very quckly. This is a good example of *pacing and play pyscology* which distracts the player from realizing the linearity of the game.

Non-Linear structure: In a non-linear level, the order of gameplay actions is mostly left to the player and therefore goes hand in hand with interactivity. This is one of the reasons why non-linearity is linked to sandbox design and to emergent gameplay. Non-linear games are rare and the closest game types with features of non-linearity are multiplayer games. A good example of this are games where a physics system allow players to manipulate their environment on their own terms, but within the restraints of the physics system.

The disadvantage of non-linear level is that unforeseen player actions or tactics may "break" the game in unforeseen ways, possibly by allowing some players to dominate others in an unacceptable way, or by finding loopholes in gameplay logic that allow the player too much power [1]. The upside however, is the satisfaction and feeling of absolute control which they get from the gameplay experience.

The methodologies of level design allow proper structuring of a level well in advance of it creation and makes prototyping easy. Potential problems can be found early, and the level creation process can be scheduled easier [1]. The methodologies includes; Annotated Maps, Concept Art and Flow Charts, and are classed under the three major parts of **stages of level design process.**

2.6 Stages of Level Design Process.

"Ideas are easy, production is hard." What matters is the ability to build and complete a high quality, compelling, marketable game.

Ellen Guon Beeman,

Producer, Monolith Productions

Unless a game is very small, it is not possible to create a complete level design as experience has shown that large games must be designed and constructed in an iterative process, with repeated playtesting and tuning, and occasional modifications to the design, throughout development. However, not all parts of the design process can be revisited. Some, such as the choice of concept, audience, and genre, should be decided once at the beginning and should not change thereafter. The process is therefore divided into three major stages - Concept, Elaboration and Tuning stage – each of which includes a number of design tasks.

2.6.1 Concept Stage

A concept is a general idea of how one intends to entertain someone through gameplay and, at a deeper level, why one believes it will be a compelling experience [3]. Ideas are formed from a variety of things ranging from real life experiences and images to sensory inputs. Turning an idea into a game is what level designers do. They take ephemeral things and give them form. This means that a good designer must learn how to distill ideas into a solid, definable, workable game [5].

The concept stage establishes things about the game that are so fundamental, changing them later would wreak havoc on the development process because a great deal of the work done to implement the game would have to be thrown away [3]. In order to design a good and viable concept, one need factor in elements that would influence the outlook of the entire game. These include, the game genre, the demography (target audience), and the role to be played by the player amongst others.

The thought-out concept is put in a well-organized written format referred to as a high *concept document.* Another key component in a high concept document is a storyboard, which comprises of images relating to those that would be incorporated into the game.

2.6.2 Elaboration Stage

At this point, design work begins to move from the general to the specific; from the theoretical to the concrete. [3] This transition begins by putting together a level design document, designing the game mechanics, white-blocking and then prototyping the game.

- Level Design Document: This document defines how and what the game world would look as well as the challenges and interactivity within the game. Traditional level designers would always start by sketching on paper then rendering them on a level design editor engine. A level design document would contain annotated maps, flow boards/level progression, and content/obstacle placement amongst other things. Another inclusion in a level design document are partis. Partis are meant to be sketches, and therefore will lack measurement [6]. The key to a level designer's parti is to sketch gameplay ideas as spatial diagrams [6].

- Designing The Game Mechanics: Although the level design document would show the challenges in a game, it wouldn't go into the detail about the mechanics upon which the challenges and objectives of the game are achieved. In other words, it becomes essential to clearly define the core game mechanics to be included in the primary gameplay mode and how they create challenges and implement actions. For example, if a sport game is being designed, features athletic characteristics such as speed, strength, acceleration, accuracy etc would be given consideration thus prompting questions like, how does a player get more speed and when does it become impossible to control etc. In a puzzle game such as Tetris, designers have to figure out how to make one level different from the next, therefore looking beyond the manipulation of symbols but also increasing difficulty level by chipping in unusual symbols while constraining the player's decision to a given time. Depending on the number of teams available for a game production, the role of designing a game mechanism is sometimes juggled between the programmer and the level designer. In some cases, an independent team – referred to as core-mechanics – is created specifically for this purpose

- White-blocking: is when a level designer creates a level out of simple geometry, most often white or textured blocks (thus the name), to test whether levels accomplish the gameplay goals he or she wants [6]. For example, every game level should have an estimated time of completion and a way of knowing just how long such completion would take is by white-blocking. White-blocking helps with other types of measurement before prototyping, measurements such as analyzing the dimension of objects best suited for the game, testing collision around the character, stretching the engine limit to know how much geometry can be used before lagging sets in etc.

- Prototyping: Video games must be prototyped before they can be built for real, and they must be tested at every step along the way. Each new idea must be constructed and tried out, preferably in a quick-and-dirty fashion first, before it is incorporated into the completed product [3]. This is carried out on game engines such as Unreal Engine, Cryengine and Unity3D to mention but a few. The prototyping phase involves building, testing and iterating, a continual cycle which only stops upon completion of the actual game. Iteration in this phase requires dexterity in asset usage, as explained by Kremer when he said gameplay can be replicated over several iterations without changing the assets needed, thus resulting minimizinig cost on gameplay.

2.6.3 Tuning Stage

The transition from the concept to the elaboration stage means that the game concept has been locked down and its foundation, firmly built. Design work enters the tuning stage, during which little adjustments to the levels and core mechanics of the game can be made, provided no new features are introduced. This stage, more than any other, is what makes the difference between a merely good game and a truly great one as it involves tuning and polishing the game until perfection is attained.

Polishing is a subtractive process, not additive, hence involving several removals of earlier production which perhaps are not considered fit for the level any longer.

3 UNREAL DEVELOPMENT KIT

Just as there are a great number of level design methodologies, there are also many game engines that game designers can choose from [6]. The focus in this case is the Unreal engine whose engine-specific factors would be addressed in relation to how it influences level design. The Unreal Development Kit is a free version of Epic Games' Unreal Engine 3 with an addition of a complete toolset. It is a popular engine thanks to its powerful level editor and advanced lighting capabilities. The level editor allows for the quick creation of levels within the engine itself with UDK's binary space partitioning (BSP) brushes [6]. Another proprietary feature of Unreal's BSP brushes is a default texture setting which results into nicely textured surfaces therefore lowering the need to apply textures using the UV unwrapping functions.

In this chapter, the term "engine" and "toolkit" would be used interchangeably as the UDK comprises of both features. This chapter would introduce different aspects of the Unreal Development Kit in relation to level design. However, due to enormous size of the Unreal Toolkit, individual sections are allocated to discuss specific aspects with respect to level design. The section starts with the introduction of the Unreal Editor which appears to be the backbone of the entire toolkit.

3.1 UnrealEd

Unreal Editor is comprised of a great deal of tools, some more complex than others

Zak Parrish (2009)

Unreal Engine 3 contains a suite of tools used to create and edit levels, import and organize art assets, create specialized content assets, and much more. The cornerstone of these tools is Unreal Editor, or UnrealEd. Unreal Editor is comprised of a level editor, a collection of browsers, and a set of specialized tools [8].

The UnrealEd is so useful that virtually every developer working on a project will spend - at least part of their - time working inside Unreal Editor. Level designers obviously will use the level editor to create environments and worlds for the game to take place in. Content creators and 3D artists/ animators will import their assets and use the specialized tools to set them up for use in the game. Programmers will use test levels to ensure their gameplay code works as expected and may be tasked with extending the editor's functionality or adding new tools [8].

The UnrealEd comprises of a Level editor, a Material editor, a Matinee editor, a Kismet editor, a Cascade, a Sound editor and more [9].

3.1.1 Unreal Level Editor

The Level editor is where developers design all 3D/2D environments by using a GUI (graphical user interface) to either drag and drop a model into the scene, move them around, orientate, scale group them etc. The Level editor works based on real-time preview, in other words, it allows a designer see every element of change within the 3D environment instantanously. This allows developers to play their levels from a player's perspective inside the editor with all the sounds, physics and animations working [9].

3.1.2 Kismet

Kismet is a virtual scripting system one can use to create complex scripted sequences quickly and easily, with surprisingly little programming knowledge [10]. Virtually every interesting thing a game needs to do during gameplay would use kismet in one way or the other. This includes tasks as easy as opening a door and as complex as launching just the right event when a player holds a specific object at a critical location in the level [10]. Kismet is the backbone of level interactivity as it provides a way for artists (the non-programming types, that is) to create complicated scripts to power the level that they have designed.

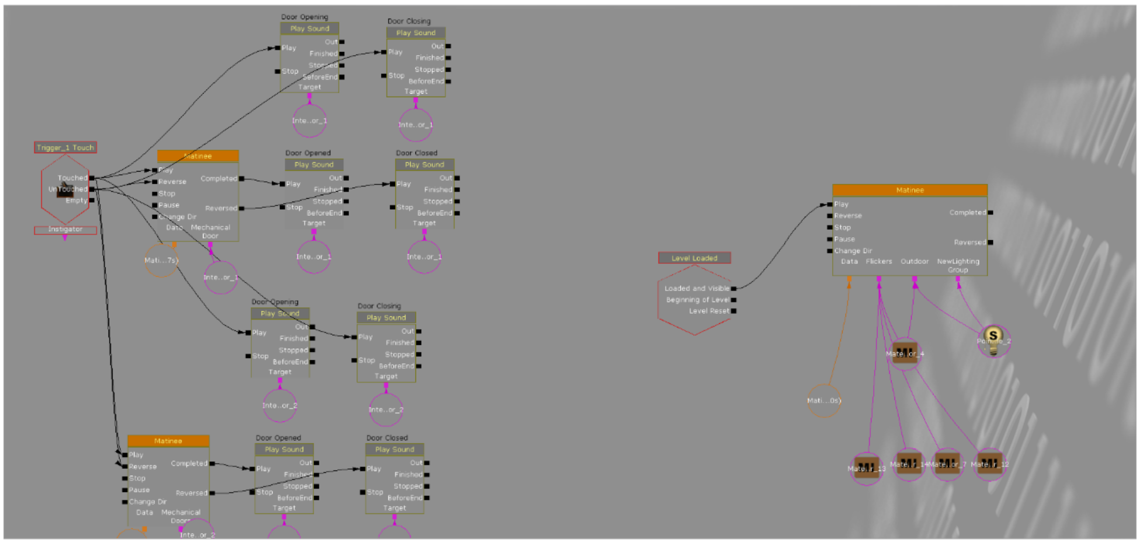


Figure 2. Kismet editor with a simple nodal program to open/close a door.

As seen in the above figure, the kismet sequences are done on a canvas where events, actions, objects, variables and matinee may be created and dropped.

3.1.3 Matinee Editor

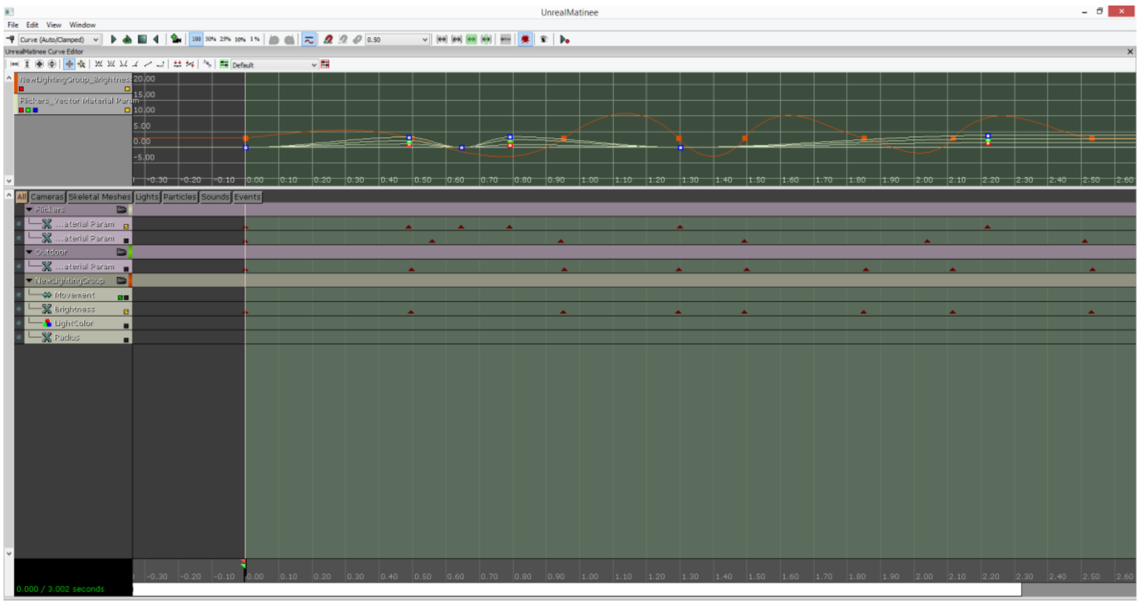
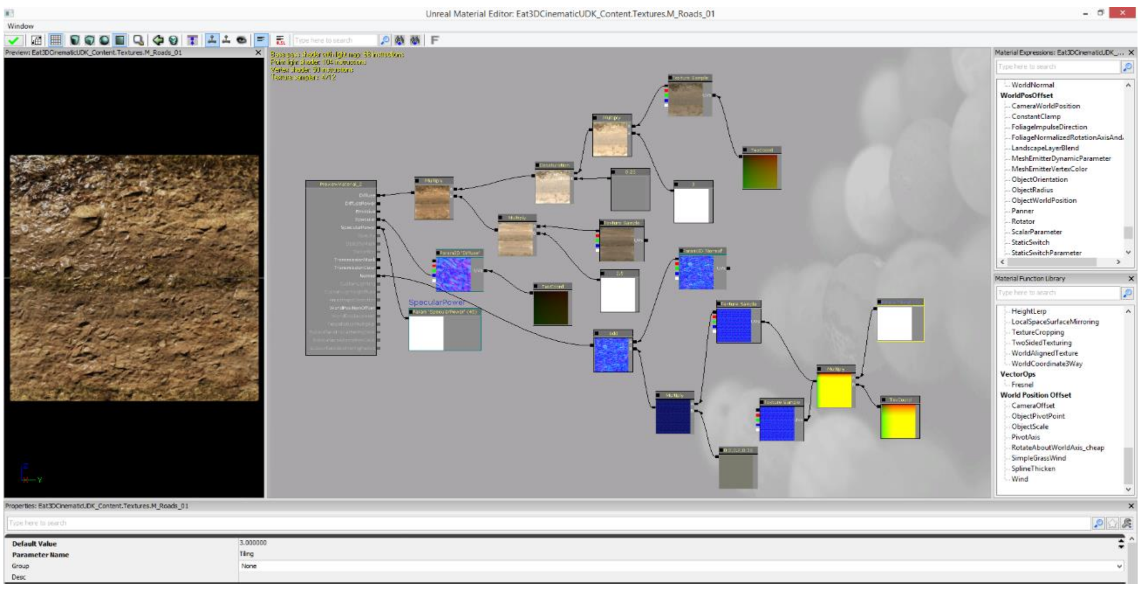


Figure 3. Matinee editor with movement graph tracks.

Matinee is integrated closely with Kismet [11]. It appears as a type of Action within Kismet and playback is started by connecting one of its inputs to some kind of event in the level [11]. Unreal Matinee is used to create motion by using keyframes, positions and properties of Actors within a scene over a given time. Its function ranges from authoring cinematic sequences within a game, creating per-level animations of moving objects (such as platforms, doors and elevators, spinning wheels), light flashes and camera effect, to complex animations (like having a flythrough of an entire level before it begins).

3.1.4 Material System

A material (in UDK) is a set of algorithm which defines the nature/appearance of any three dimensional surface and how it reacts with light. The Material Editor provides a platform for material authoring using visual node-based graphs. The nodes are connected to different channels within of a material. Depending on what the expected result is, the channels allows a level designer to control glossiness, transparency, bumpiness, shininess, color, reflectivity, and several other features of the 3D surface.



Default Value	3.000000
Parameter Name	Tiling
Group	None
Desc	

Figure 4. A wet rock surface created using unreal material system.

Material Expression are the fundamental building blocks used to create material [10]. Each expression contains a certain functionality such as providing an interface to a texture, adding two values or vectors or modifying vector coordinates. By combining all these expressions into an elaborate network, many interesting effects can be created resulting into a more striking visual for levels.

3.1.5 Cascade Particle System Editor

A particle system is a collection of one or more emitters, each of which has a particular arrangement of rules,allowing for easy creation of a wide variety of visual effects that require complex, repetitive, and/or predictable behaviour [12].

This system is built from two parts namely: the emitter and the particle. The emitter is the logic of the particle system. It emits particles at a specified location and determines the behavior pattern of those particles once they are created, such as the effects of gravity, and how they move. A particle is a single object that the emitter generates, for example a single rain droplet of a rain effect [9].

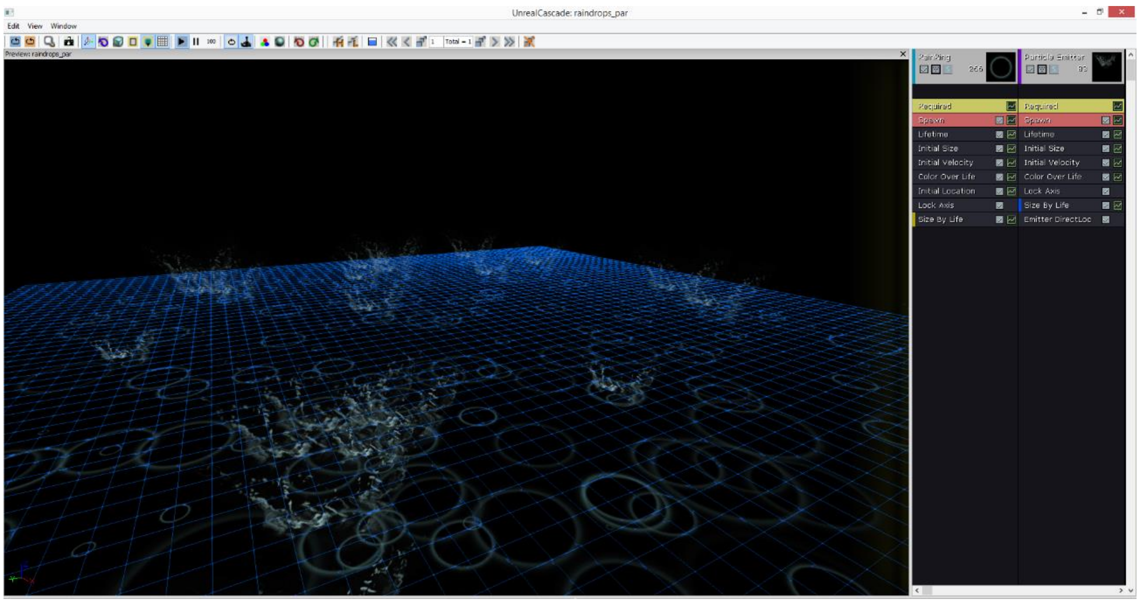


Figure 5. Raindrops, splashes and ripples effect created with particle system.

Each emitter within a particle system can be designed to behave in an independent fashion thereby creating different aspects of a desired effect. For example, a bonfire would require flames, heat distortion, smoke, falling sparks and a crackling sound, all of which can be assigned to seperate emitters yet collectively makeup one particle system.

From the level designer’s standpoint, the heart of the particle special effects creation system in Unreal Editor is the intuitive Cascade editing system [12].

The editor provides complete control over every aspect of particle modification and fine-tuning, including emitter creation, the addition of modules, the tweaking of properties – such as Lifetime, size by life, initial velocity,initial size, etc - , and visual evaluation [12].

3.1.6 Terrain System

Unreal Engine 3 supports a flexible terrain system that provides a wide variety of visual styles and uses [13]. The system makes creation of landscapes easy through the provision of *heightmaps* (which are useful for depicting hills, mountains, plateaus, roads to mention but a few), *Material System* (responsible for applying textures to the terrain) and other features – such as *foliage and multi-decoration system* – complementing the design of an outdoor scene. A terrain mesh is basically created using one of two methods: direct hand-painting on the terrain mesh by using the heightmaps, or importing an externally created terrain map (with exterior applications such as Terragen, World Machine etc).

To prevent unnecessary calculation and rendering of details outside the field of view of a camera, the geometry of a terrain actor is dynamically tessellated during gameplay, thus allowing the number of polygon and detail level to constantly change.

One of the challenges that level designers face is choosing the appropriate layout and resolution of this terrain mesh in order to provide the best visual quality versus performance setting [13]. To create a balance, unreal terrain editor provides a maximum world size of 512k x 512k (524288 x 524288) Unreal Units but recommends creating nothing larger than 1024 x 1024 patches because, not only is this the most effective set of values for patch size but also because it is the number of patches required to obtain the best balance between terrain detail and rendering performance.

A handpainted terrain mesh utilizes two features from the UnrealEd namely; TerrainMaterial and Layers, weight maps.

- TerrainMaterial and Layers: TerrainMaterial are divided into groups referred to as layers. Layers provide the level designer with a way to setup groups of materials that can be exposed on the terrain and blended together based on a variety of circumstances such as the elevation of the terrain or its slope along a hill or mountain [10].

- Weight Maps: Terrain uses weight maps to determine the exact placement of each material on a terrain surface [10].

- Foliage and Deco layers: Amongst the methods of making a terrain appear believable is to decorate it with realistic objects such as bushes, rocks, trees and other items found in an open landscape. In the unreal terrain system, this can be handled through the use of Foliage and Deco layers [10].

3.1.7 Brushes And Volumes

Brushes

Brushes are the most essential tool when creating a base layout and geometry for any level. These brushes can also be invaluable in ”prototyping” levels: quickly previewing what they will look like and how they will play, without creating all their individual assets [10].

BSP brushes (Binary Space Partition) allows level designers to quickly create and lay out surfaces such as floors, walls, and ceilings which are lined up with each other without having to worry about using any other software [14]. Another vital feature of BSP brushes is that it allows created geometries to be exported as static meshes – into other 3D editors – thereby allowing designers the ability to match a model to the BSP proportion created in Unreal Engine. Besides being used for modeling, BSP brushes serve as triggers, collisions and volumes, all of which a level designer requires to instantiate an event, set barriers, simulate actions amonsgt other things.

The outcome of a level's complexity is often dependent on the judicious usage of the many features of brushes. As of the release of Unreal Engine 3.0 users can create a level in an additive or subtractive manner, rather than being limited only to the subtractive levels as they were in previous generations of the engine [10].

Volumes

Volumes are three-dimensional spaces programmed to be aware of when an object (such as a player) has entered them [12]. As a result, these spaces affect an object in a variety of ways thereby enhancing level interactivity, from triggering sound cues to altering the laws of physics within the engine.

Volumes can also provide a simple means to handle level streaming,which allows them to replace the older concept of zoning that was used for level optimization in versions of the Unreal Engine prior to 3 [12]. Levels in Unreal engine are – by default – encased in one massive cubical physics volume refered to as *DefaultPhysicsVolume*. A variation in physics – such as levitation as a result of some force field, bouyancy when swimming etc – different from the *DefaultPhysicVolume* would require the creation of a new volume (within the default volume) using the builder brush.

There are several volumes in UE3 which serve different purposes necessary for quality level design . These volumes include; Blockingvolume, DynamicBlockingVolume, DynamicTriggerVolumes, LadderVolume, GravityVolume, LevelStreamingVolume, PhysicsVolume to mention but a few.

4 THE MUDSKIPPER FAMILY – CASE STUDY

In this chapter, an overview of the game – The Mudskipper Family – is discussed alongside the weighted relevance of different parts of the Unreal Development Kit all through the phases of design. Section 4.1 introduces the concept of the game, its story, characters and mechanics. The second section, Section 4.2, discusses the implementation phase; level sketches, white-blocking and prototyping amongst others. At the end of the chapter, a section is devoted to summarize the chapter’s content.

4.1 Concept

“A proof of concept is not a game at all”

Ernest Adams (2003)

4.1.1 Story

Like any good level design in AAA games, the concept of the game needs to be understood in order to reflect in the artistic or architectural structure of the levels. A game like God of War has a story built on ancient greek mythology hence its environment was designed to bring life to the story. The Titans, Mount Olympus, The Underworld (Depth of Hades) and many more where represented in a very complimentary fashion that suits the game while. In other words, the story behind any game is just as important as its visual interpretation, coherence and gameplay.

The Mudskipper Family, like the above mentioned game, was equally born out of ideas which became formalized and documented in a bid to ensure comprehension and make meaningful prioritization.

The Mudskipper Family is a tactical adventure-based, third person shooter game with the ultimate goal of liberating a captured mudskipper. The game is set in the lush Caribbean island of Curacao, peaceful in all of its essence. The Mudskipper Family is made up of a family of five, each possessing different unique abilities to help avoid predators while in the open. A player has access to only one of four characters at any given time during the game, the wisdom behind this was to avoid overcrowding of the screen which might inadvertently confuse/affect the player's selection.

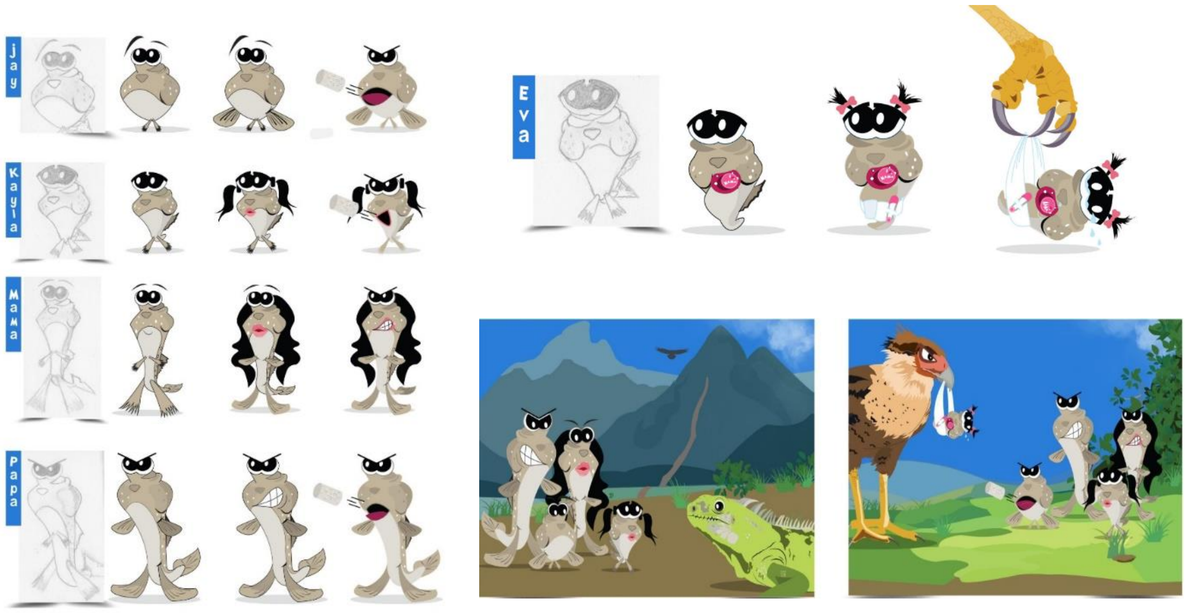


Figure 6. The Mudskipper Family concept art.

4.1.2 Mechanics

The core mechanics of the game revolve around five features;

- overcoming obstacles by defeating minion bots
- refilling the health bar at checkpoints to keep the character hydrated
- solving patterns to escape mazes and defeating end bosses in level and
- knowing the abilities of each character and how they come handy during gameplay
- Collecting flies which accumulates and yield an extra life.

All characters by default have a uniform defence mechanism against enemies. They all spit sand (like real mudskippers do) with the same hit point but different distance of projection, some farther than others. Each level begins with **Papa** as the main character and players can toggle the character until they find the desired character they choose to use. Amongst his features is having the highest jumps, spitting sand the farthest and squirting water (a unique ability which helps destroys fractured static meshes).

Mama spits sand and jumps just as high as high as the Papa. Her unique feature is her ability to glide in mid-air.

Jay is a sturdy character with the least jumps. What he lacks in jumps, he makes up for in strength. Two features proprietary to Jay is stomping – sending seismic wave around a given radius which kills any enemy cause within it – and pushing objects.

Kayla, the fourth of the bunch shares the same jump power as **Mama** and uses her small stature for either passing through obstacles that cannot be broken.

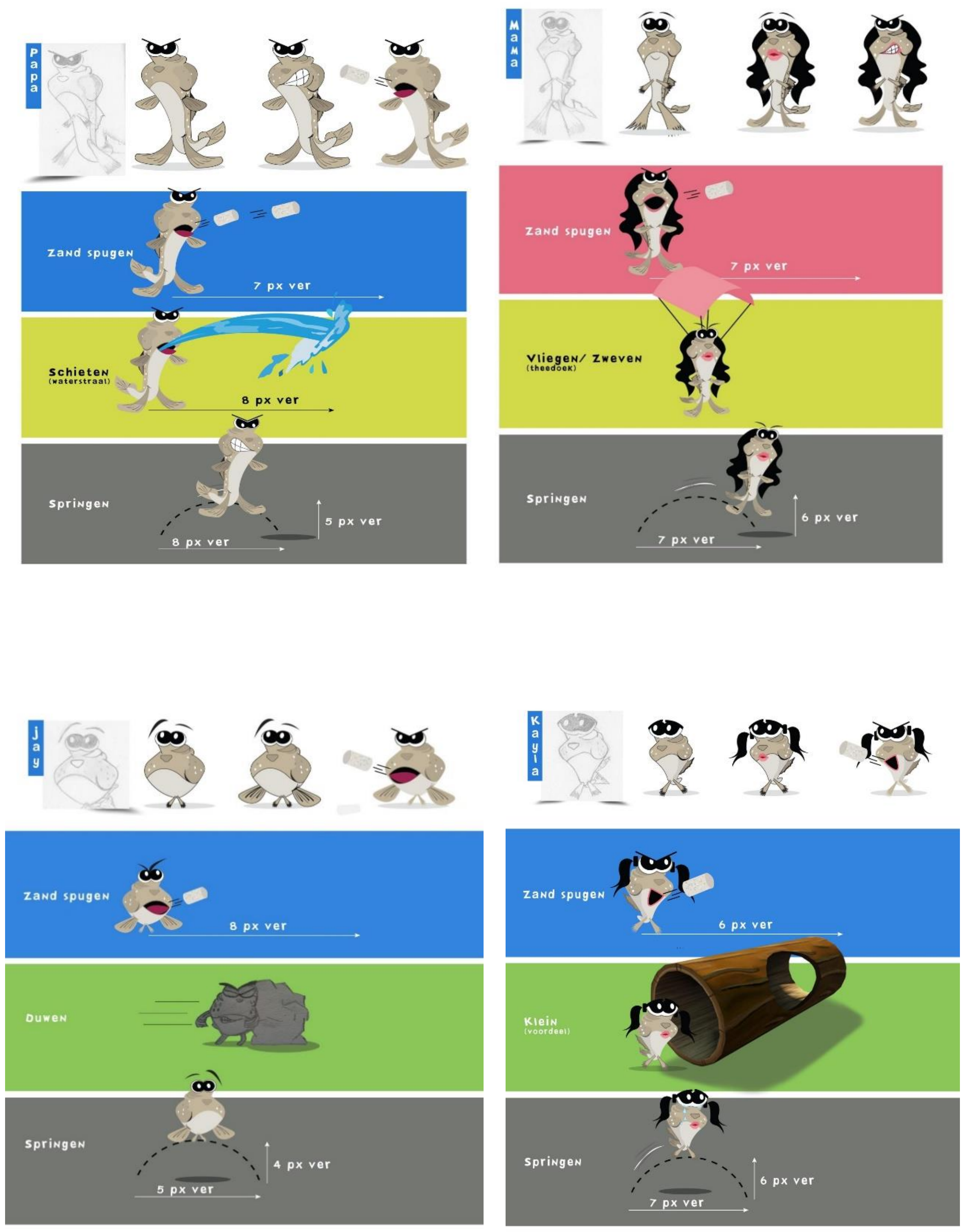


Figure 7. A representation of the abilities of usable characters.

Level bosses were designed to be intimidating in size whilst the degree of difficult increases as players progresses. Their mechanics was one inspired by the games; Beach Buggy Blitz, Rayman and Kung Fu Panda, causing them to share certain common features. One major similarity is the boss's confined movement to a fixed radius upon encounter. This way, the player has the advantage of mobility around the boss – who can only rotate on his own axis and launch attack or counter attacks from there. Another adopted mechanic is the frenzy state which the bosses goes into when their health becomes low.

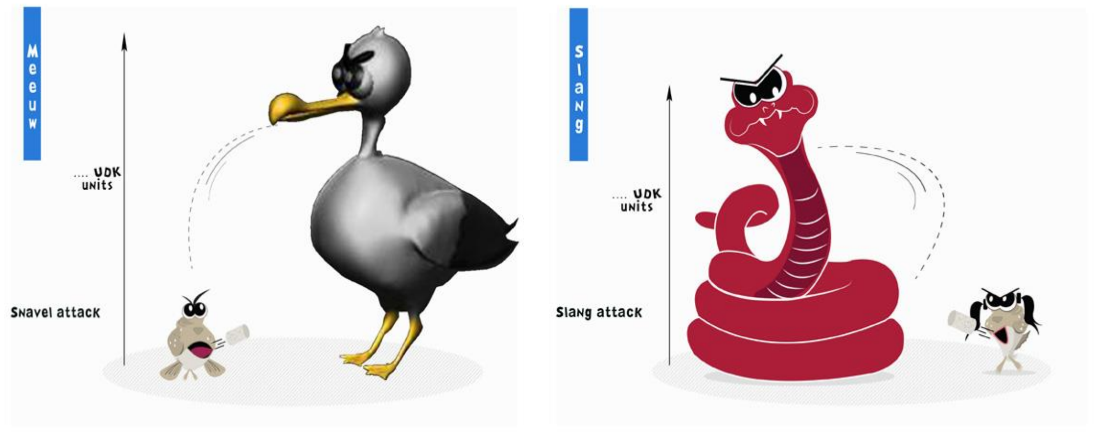


Figure 8. Character dimensions and attack/defence mechanism.

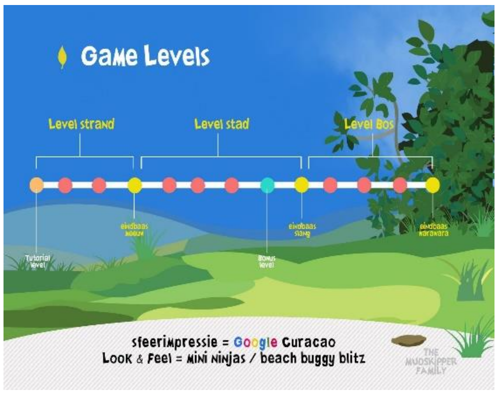
4.1.3 Level Design Document

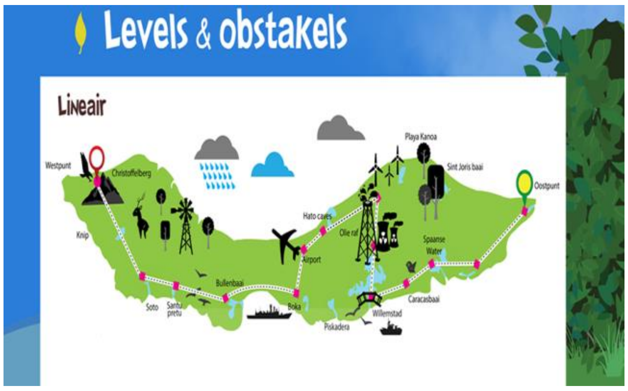
The game comprises of three map packs - beach, urban and jungle - each of which contains four levels.

- The beach map pack was designed to portray the coast lines of Curacao. It starts with a tutorial level and followed by three other levels namely; Nieuwpoort, Spaanse Water, Caracas Bai.

- The urban map pack showcases the beautiful metropolitan region of Curacao allowing players to transition through the oil-rich Willemstad – Windmolen – Olievelden – Boka.

- The jungle map pack starts with Bullenbaai – Santa Pretu – Soto – Christoffelberg.





From the left, Figure 9[a]. An image showing map packs, levels and linear transition route within the game. Figure 9[b]. A graphical representation of the features in each level of the game.

4.1.4 Game and Level Requirements

A playable game can be attained with the minimum of one level, therefore , this section would detail the requirements needed in the creation of a playable prototype level in the Mudskipper Family game. Additionally, this section would also describe the game mechanics requirement derived from the concept document. The requirements listed here would potray a simple level to show off the game mechanics and design. To begin with, a list of content needed to build a simple prototype level is created in Table 4.1.4a and 4.1.4b while Table 4.1.4c lists the requirement related to the game mechanism and interactivity.

Table 1. L01: Minimum required asset for a level in the Mudskipper Family game.

ID	Requirement Description	Priority
L01.1	The level contains rocks	H
L01.2	The level contains tikki huts	H
L01.3	The level contains fishing nets	L
L01.4	The level contain barrels	H
L01.5	The level contain surf boards	H
L01.6	The level contain tug-boats	H
L01.7	The level contain wooden platforms	H
L01.8	The level contain trees and plants	H
L01.9	The level contain hollow trunks	L
L01.10	The level contain fences	H

Table 2**.** L02 : General requirements for protyping the Mudskipper Family game.

ID	Requirement Description	Priority
L02.1	The level includes a terrain	H
L02.4	The level should include light sources for illumunation of the geometries	H
L02.5	The level includes a water body	M
L02.10	The level is completed only when the finish line is crossed	H
L02.11	A map pack is finished only after defeating the end boss	H
L02.12	Upon completion of a level or map pack, level-streaming must occur to initiate the next level or map pack.	H

Table 3. GM01: Game mechanics requirement for a prototype.

ID	Requirement Description	Priority
GM01.1	Level contains spawn/respawn points which doubles as checkpoints	H
GM01.2	When a character dies, he respawns from the closest spawn point.	H
GM01.3	Charaters rehydrate from checkpoints	
GM01.4	Terrains are rigged with booby traps and moving objects	H
GM01.5	Characters should have 3 health bars and lose one either as a result of dehydration, being killed by enemy bots or falling into a trap.	H
GM01.6	The terrain is created to support AI pathfinding enemy bots should be able to stalk and harm a player	H
GM01.7	Characters share a single hydration meter which depletes Overtime unless replenished from a checkpoint	H
GM01.8	End bosses become active when characters are within a certain proximity	H

4.2 Implementation

"Bad level design can ruin a good game."

Kremers Rudolf (2009)

Having introduced The Mudskipper Family game and establish its core mechanic, this sub-section details the steps taken into creating some of the levels in the game as far as gameplay is concerned. Rollings and Adams were right in saying that there is no one way to design levels [3], the approach during this phase began with sketching and crude modelling to generate new building ideas. Iteration here required consistent fine tuning of the levels all through development. This process of iterative refinement is not an excuse to introduce major changes into the game late in its development, nor to tweak it endlessly without ever declaring it finished [6].

4.2.1 Sketches

University of Washington Professor Emeritus Francis Ching said that drawing both "invigorates seeing" [15] and "stimulates the imagination" [15]. The importance of this non-digital prototype– irrespective of how primitive – cannot be overemphasized. It not only help a level designer learn to create spatial layouts but also evoking emotions through the gamespace.

"...make sure you really intimately know the game itself and all of its subtlety so that you can position that knowledge." Jim Brown, the Lead Level Designer at Epic Games"

In view of the scholarly position above, a lot of games were played to help understand how to combine gameplay beats, level difficulty, story-telling narratives, pacing and flow into a cohesive single-player level design experience. Added to this, is the challenge of depicting unique characteristics for which Curacao is known, thus resulting to endless researches about Curacao, its terrain, renown places and identifiers that could be incorporated into the game amongst others. Taking level 4 (Caracas Bai) as an example, the topography was modelled after the real beach and gameplay elements were inspired by sightings common along the bay .

Sections were used in tandem with the plan drawings to describe three dimensional space as shown in the figure 10. According to architectural writer Matthew Frederick, “Good designers work back and forth between plans and sections, allowing each to inform the other” [16], such was the process for the larger part of this stage of design. Some elements such as speed boosters and invincibility were added in earlier sketches – to enhance gameplay – but were removed overtime.

Professional level designers adopt the use of graph papers which they find especially useful for figuring out the proportions of objects in sketches. However, the sketches made for The Mudskipper Family game were more of partis, in that they lacked measurement and included details of the game mechanics. This way ideas were formed quickly before spending time to plan measured versions of the designs.

The sketches below are of two various types, axiometric and top-down.

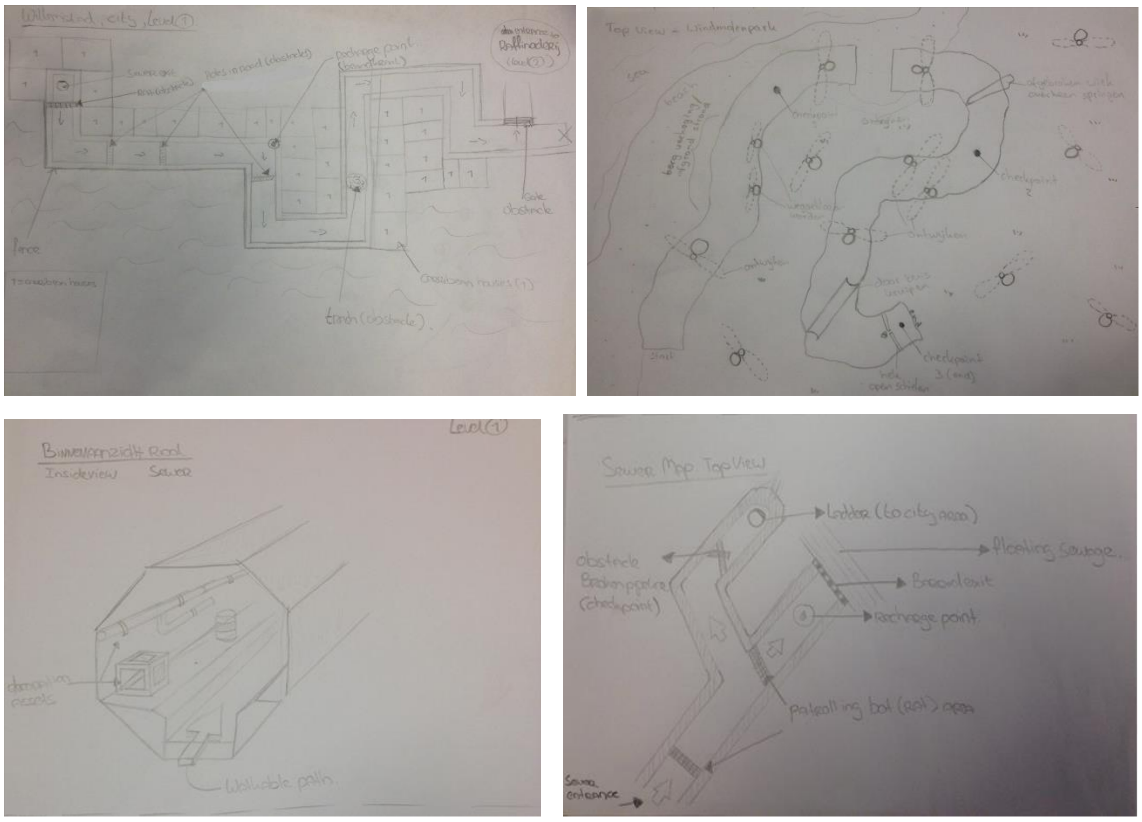


Figure 10. From the topmost left in an anticlockwise fashion [a] a sketch of williamstad [b] The sewer level [c] a top-down view of a section of the sewer [d] Windmolen park.

4.2.2 White-blocking

As earlier explained in the chapter 2, white-blocking helps level designers to determine the what gameplay measurement should be. The Mudskipper Family game consists of 12 levels, all of which had to be white-blocked for measurement as well as stress-testing purpose. White-blocking of the game was done for two purposes ;

- To test the stress-level of the engine
- To simulate the colliders to be used in the eventual design.

The first purpose (stress-testing) revealed two of the major problems a level designer would encounter in the course of any game development when using Unreal Engine 3. The evalaution are as follows;

- Geometries are the simplest ways to whiteblock a level and the Unreal engine 3 provides numerous tools to suit this purpose. During the white-blocking of the sewer level, BSP brushes were used to create a maze sewer. The result of this was an instantaneous and repititive crash of the engine upon loadig the level. It was discovered that although Epic games – the makers of Unreal engine – never warned against the overuse of brushes, the problem, if undetected in the early stage of design, would have marred the entire project.

- Another problem observed during white-blocking is the effect of lighting and how its calculation would result to a having a slow build and a significantly large file.

Bearing in mind that white-blocking is the first stage of digital prototyping, when simulating colliders, the goal is to test player's interactivity with physical objects, some of which might be retained in the final prototype. In several levels scattered within the three map packs of the game, colliders were introduced to act as

- Kill or damage volumes which either kills the character or deplete its health. One of the screenshots in figure 12 shows a waterlogged ditch, if a character falls into it, a collider is set to kill it after a certain depth, this way, an infinite fall would be avoided .
- Blocking volumes to define boundaries and avoid falling off the game world.

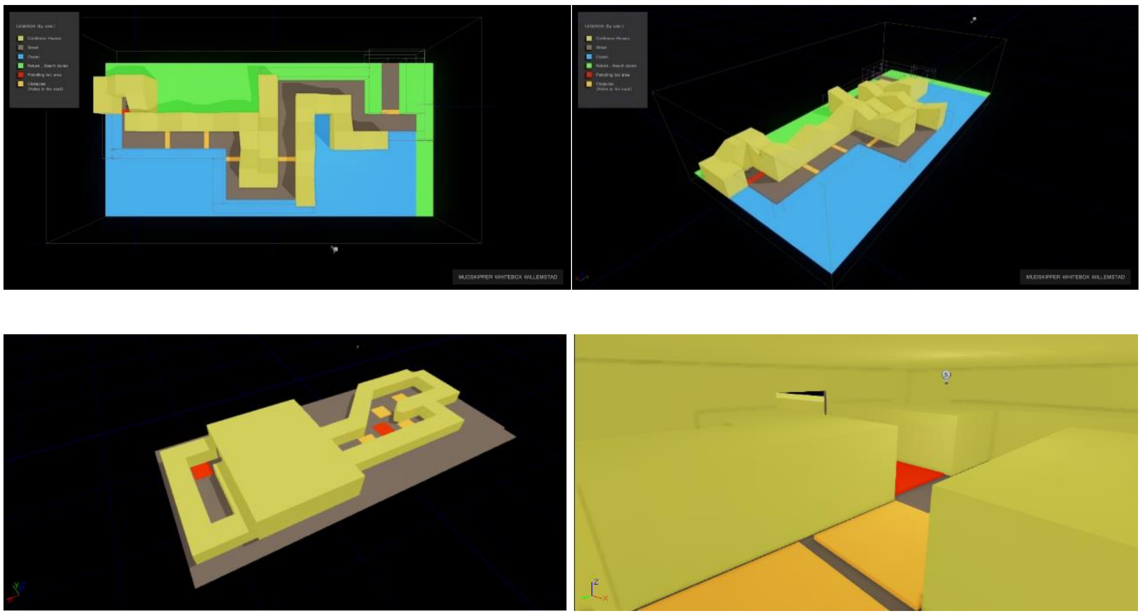


Figure 11. From the topmost right, moving counter-clockwise [a] and [b] shows white-blocking of williamstad. [c] and [d] Shows the white-blocking of Santa Pretu.

4.2.3 Prototyping

This section would outline the prototyping process upon which a prototype is created whilst adhering to the design document in chapter 4.1.3. As mentioned in the earlier chapters, prototyping is about iteratively refining basic sketches or white-block prototypes into a final product. Having concretized the textual based sketches into white-blocks and using placeholders as temporary assets listed for the level requirements, this phase of design showcases the power and relevance of UnrealEd with respect to the actualization of the game levels. In the current state of the prototype, a lot of emphasis was placed on the outlook of the level, the challenges (enemies and how they act or react), obstacles (puzzles) and the character's interaction with the game world.

Terrains were created in the Terrain Editor and populated with vegetations - which serves as good hideouts for enemy bots – through the foliage system. The materials applied on the terrain were modified in Unreal Material System, thus allowing the blending of several materials to create something new. One of the most important aspect to be considered here is pacing. A game such as this (with an age demography of 7 and below) would be expected to consist of levels that are considerably short and easy, lest it bores the player. To implement this , every level was created to last between 3 – 4 minutes without any gameplay element or interaction whatsoever. The terrains were adjusted until this time requirement was met after which materials were added. The inclusion of other elements ranging from dynamic objects to enemy bots required a tolerance of between 5 – 7 minutes for completion without boring the player.

Using the foliage system, vegetations were added into the level. The system was especiially useful in both the beach and jungle level. With the occlusion feature turned on, vegetations farther away are shown unrendered thereby conserving memory. The sparse vegetation in the level were also as a result of the foliage system's scale/proportion slider.

Moving objects – such as platforms, rocks or traps – were created by converting the assets or static meshes into InterpActors, creating a movement track for them and then connecting both in Kismet. This way, the trigger button allocated to the asset is turned on based on the players proximity, which in turn initiates matinee and is executed by the kismet. Figure 12 shows a movement chain trap which can not be overcome save except through careful study by the player. The same feature was implemented to create floating platforms which sinks if a player stays on it for too long , amongst other game mechanics.

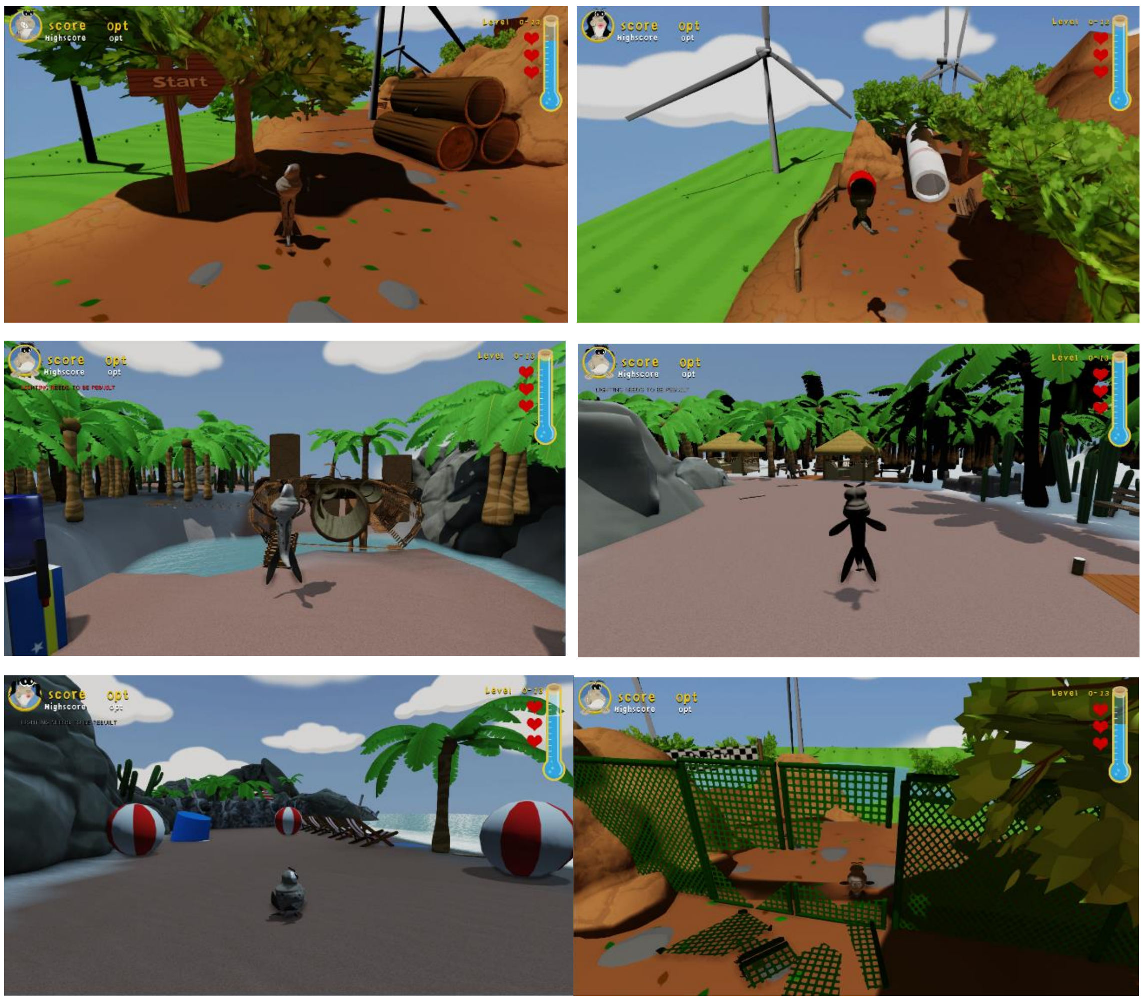


Figure 12. Screenshots showing prototyped levels and their visual standards.

The movement of melee enemy units are created using both the unreal scripts as well as kismet. Enemies such as sewer rats and snakes operates on a script known as flocking. This script is designed to enable crowd management, this way, the agents (or characters to whom the script is assigned) do not appear in disarray. The script ensures the enemy bots not only move together without colliding with one another, but also ensures each agent move together in similar fashion like a flock of bird hence the script name. Added to this script is a network of kismet features responsible for character transition in state. The kismet's AI guides each state (idle, attack and retreat) assumed by the enemy bot, this way they only attack when a player is within a certain proximity and retreat when he is beyond reach.

The vividness of the level implementation can not be fully captured in texts as several elements such as particle effect for the creation of butterflies, boulders using physics, working with shaders to show movement and dynamic reflection of water, level streaming amongst others were designed to add to the looks and feel of the game's overview.

4.2.4 Polishing

The focus at this stage was about creating enjoyable and highly interactive ready-to-be-played levels. To understand how the levels translate to fun from a players' perspective, we (myself and three other level designers) invited 10 students - not older than 7 years old - from a neigburing primary school to our GameLab in Han University. We brought the kids into the laboratory one at a time and observed them play the game from behind a screen while ensuring we had little to no interaction with them. They had to start with the tutorial level and gradually work their way up to the end thereby completing three separate levels. The testers were later asked to review some levels by answering questions such as

- Was the tutorial level easy to understand?
- How difficult were the levels on a scale of 1 – 3? (with 3 being difficult)
- Was the game fun or boring?
- Were the levels too long or short
- What did you enjoy/dislike about the game?
- Would you play it again (with all complete levels)?

The information got from this questionnaires coupled with the results of the observation was used to improve upon several other levels. Players would usually play a game in ways other than it was design, a situation which often lead to breaking the game or – as discovered in this case – detecting bugs. Recreation of blocking volumes , designing collision boundaries for fractured static meshes, redesigning failed triggers on kismet, aligning the player's character to the terrain (to prevent sinking or floating), alignment of vegetations to the ground etc were amongst the bugs detected and fixed.

5 CONCLUSION

A toolkit the size of the UDK requires a great deal of training before it can be effectively used. It provides a unique platform for level designers the likes of which is unprecendeted. This learning process of building prototypes on the unreal engine particularly from a level designers perspective serves as an eye-opener to why the Unreal engine remains unequivocal in terms of design, user-friendliness and high fidelity products. A lot effort was put into research of the engine usage as well as getting acquinted with the processes of game development and where to draw the line between the tasks of a level designer and an art.

The Mudskipper Family game was intended as both a PC and mobile game. However, owing to time constraint, the final product was only available on PC. This thesis does not discuss game development as a whole nor does it discuss aspects of game development/design outside the sphere of level design, this is because it makes a case study of my role in the project and would hopefully serve as a good academic reference for future level designers (using the unreal engine 3).

REFERENCES

[1] Kremers, R. (2009). Level Design. Wellesley, MA.: A.K. Peters.

[2] Adams, E. (2003). Break Into The Game Industry. Emeryville, CA.: McGraw-Hill/Osborne.

[3] Adams, E. and Rollings A. (2010). Fundamentals Of Game Design. Berkeley, CA.: New Riders.

[4] Sholler, G.(2013). Build A Game With UDK. Brimínghman, UK.: Packt.

[5] Feil, J. and Scattergood M. (2005). Beginning Game Level Design. Boston, MA.: Thomson Course Technology.

[6] Totten, C.W. (2014). An Architectural Approach To Level Design. Natick USA: Taylor and Francis.

[7] Byrne, E. (2005). Game Level Design. Hingham, Mass: Charles River Media.

[8] Unreal Editor and Tools, 2012 [ONLINE]. Available at: https://udn.epicgames.com/Three/EditorAndToolsHome.html [Accessed:10 June 2016]

[9] Thorn, A. (2012). UDK Game Development. Boston, Mass: Course Technology PTR.

[10] Busby, J., Parrish, Z. and Wilson, J. (2009). Mastering Unreal Technology (Volume 1). Indianapolis, Ind.: Sams Publications.

[11] Unreal Matinee User Guide, 2012 [ONLINE]. Available at: https://udn.epicgames.com/Three/MatineeUserGuide.html [Accessed:10 June 2016]

[12] Busby, J., Parrish, Z. and Wilson, J. (2009). Mastering Unreal Technology (Volume 1). Indianapolis, Ind.: Sams Publications.

[13] Terrain Design, Guidelines and Information, 2012 [ONLINE]. Available at: http://udn.epicgames.com/Three/TerrainDesign.html [Accessed:10 June 2016]

[14] Mooney, T. (2012). Unreal Development Kit Game Design Cookbook. Olton Birmingham [England]: Packt Publications.

[15] Ching. F.D. K and Juroszek, S.P. (1998). Design Drawing. New York: Van Nostrand Reinhold.

[16] Frederick, M. (2007). 101 Things I Learned In Architecture School. Cambridge, Mass.: MIT Press.