Design, Development, and Implementation of a 2D Puzzle-Platforming Adventure Game

A Senior Project Submitted to the

Department of Computer Systems Engineering Technology

of the School of Engineering, Technology, and Management at the Oregon Institute of Technology

in partial fulfillment of the requirements for the Degree of

**Bachelor of Science**

Christopher Horenstein

© June 2011

**Senior Project Approval Page**

The senior project of Author 1, Author 2, and Author 3 for the Bachelor of Science degree was accepted by the evaluation committee and the Department of Computer Systems Engineering Technology at the Oregon Institute of Technology.

COMMITTEE APPROVALS:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Jay Bockelman, Program Director (Oregon Institute of Technology)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Reviewer 2 (Oregon Institute of Technology)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Reviewer 3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Reviewer 4

**abstract**

The abstract for the project goes here. The abstract serves as an executive summary of your project. It should include a brief introduction to the project, state its relevance, describe the design features, and provide a short summary or conclusion. The abstract should be structured in three or four paragraphs and it should not be longer than a page. It is typically written after you have completed the project. (1 page)

**acknowledgements**

The acknowledgements go here. You may want to check some acknowledgement sections in books or textbooks to get an idea of the tone and content of the typical acknowledgement. It may also be a good idea to take a look at some Master’s Thesis, Doctoral Dissertations, or other Senior Projects to get a sense of the tone and language typically used in the acknowledgments section. It is typically written after you have completed the project. (1 page)

**List of Acronyms**

**XNA** XNA’s Not Acronymed

**Acronym** Definition

**Acronym** Definition

**Acronym** Definition

**Acronym** Definition

**Acronym** Definition

**Acronym** Definition

**Acronym** Definition

**Acronym** Definition

**Acronym** Definition

**Summary Table of Contents**

[Chapter 1 Introduction 8](#_Toc279749328)

[Chapter 2 Background 10](#_Toc279749329)

[Chapter 3 Functional Description 11](#_Toc279749330)

**Table of Contents**

[Chapter 1 Introduction 8](#_Toc279749288)

[Overview 8](#_Toc279749289)

[Product Description 8](#_Toc279749290)

[Existing Products 8](#_Toc279749291)

[Summary 9](#_Toc279749292)

[Report Outline 9](#_Toc279749293)

[Chapter 2 Background 10](#_Toc279749294)

[Overview 10](#_Toc279749295)

[Background Information 10](#_Toc279749296)

[State of the Art 10](#_Toc279749297)

[Summary 10](#_Toc279749298)

[Chapter 3 Functional Description 11](#_Toc279749299)

[Overview 11](#_Toc279749300)

[Block Diagram 11](#_Toc279749301)

[Summary 12](#_Toc279749302)

# Chapter 1 Introduction

## Overview

In this chapter I will describe the platforms and main idea behind Ne+, as well as describe the target market for the project and some examples of similar products already available.

## Product Description

Independent game development has been made far more accessible than ever before thanks to technologies like XNA and the Windows/Xbox 360 platforms. With these new technologies comes the ability to quickly and efficiently create video games with very small teams and even be able to sell them through Microsoft’s Xbox Live Marketplace.

In the market today independent games fall into very few categories. There is a group of games that were created in very little time and contain no new concepts, and cheap or placeholder artwork; games that were created by one or two people and did not receive polish or further development by talented individuals. Another group of independent games, arguably the most successful, develops a new or somehow unique gameplay concept, and usually displays the concept and allows the player to interact with it through various puzzles that help the player to consider the concept’s intricacies.

Ne+ contains these characteristics – it is a game that is more about the gameplay than the story or graphics, although both are ideally satisfying for the average player. It introduces a concept based entirely around different lights creating different effects on their environments that the player is able to interact with. Many different puzzles will highlight the nearly endless possibilities the concept allows for, and the tools will also create a sort of sandbox for the player to interact with, although the main game will be entirely about progressing through environments linearly.

## Existing Products

 Braid is one of the games that brought independent game development into the public’s eye. It illustrates many of the concepts that Ne+ aims to take advantage of – first and foremost, the gameplay is absolutely the heart of the game. Take everything else away, and we’re still left with the core of the experience. Ne+ aims to highlight gameplay in a similar way. It also is puzzle-based to illustrate the core concept most efficiently, which is how Ne+ presents its concepts as well. Braid also demonstrates an incredible polish, with beautiful visuals and audio, and while Ne+ artwork is not as intricate the end result relies heavily on lighting and should provide high-quality visuals and audio as well.

Figure 1: Braid

 Limbo is another game that Ne+ is similar to – it is a sidescrolling platformer like Ne+, and uses puzzles to illustrate its main gameplay concepts. Limbo experiments with visual aesthetic, ambient sounds, and minimalism, which are all secondary but necessary elements of Ne+, and elements that I am experimenting with. It is almost entirely about players using logic to progress through the game’s environments, although the length of the game is slightly lacking. Ne+ will most likely end up to be about the same length given that I am the only developer working on it, which is another similarity between the titles.

Figure 2: Limbo

## Summary

## Report Outline

This report is split into a few major sections. First, a background of the project will be given, followed by a functional description of the product, a detailed description of the product’s design and architecture, and finally test results, economics, and a summary of the report.

# Chapter 2 Background

## Overview

In this chapter, I will discuss some of the relevant background information necessary to understand the rest of the report, as well as the project as a whole. The state of the art will be addressed, and finally I will summarize the major points of the chapter.

## Background Information

To understand Ne+, it is necessary to understand the concept behind its gameplay. Video games, like all software, has the potential to be extremely dynamic in nature, providing an excellent and unique experience to every user while still adhering to some framework that provides a product’s utility in one way or another. With Ne+, the entire game is focused around the idea of different colors of lights implementing different effects on the game objects they touch. Some might even create physical entities in the game world, like a light that projects a light platform. This provides a very strong framework for endless possibilities based on creating components and letting them interact, and that concept is an important piece – the main driving force – behind the project.

## State of the Art

The state of the independent game development community is incredibly strong, which is part of the reason that Ne+ is at all possible. With games like Braid and Limbo providing very strong single player experiences from very small development teams, the state of the art is such that, with the help of open-source technologies and Microsoft’s XNA technology with platforms like Windows 7 and the Xbox 360, it is perfectly possible for a single person to create a game from start to finish in a reasonable development time as contrasted with similar tasks years ago.

## Summary

With the core concept of Ne+ being component-based and providing an excellent framework on which the rest of this report and the game itself are based on, as well as the knowledge of the current state of affairs for small development teams which are now able to do much more than they could at one point, now is the time for independent developers such as myself to embark on projects like Ne+.

# Chapter 3 Functional Description

## Overview

Ne+ has a few key elements that makes it what it is. To provide a functional overview of a video game, visuals, audio and gameplay are the key elements that must be described.

### Visuals

The visuals of the game are similar to games like *Geometry Wars*, where bright neon lights with a dark backdrop create a world in which the objects provide the only lighting. This highlights Ne+’s core gameplay element, the lights, extremely well, which is why it was deemed appropriate.

### Audio

The audio elements of Ne+ are largely electronic sounds, with slow and quiet electronic music and various sounds coming from game objects. There exists a connection between the audio and visuals beyond game actions; the lights coming from certain game objects are manipulated in accordance with the audio, so that certain game objects pulse with music to provide a more immersive aesthetic.

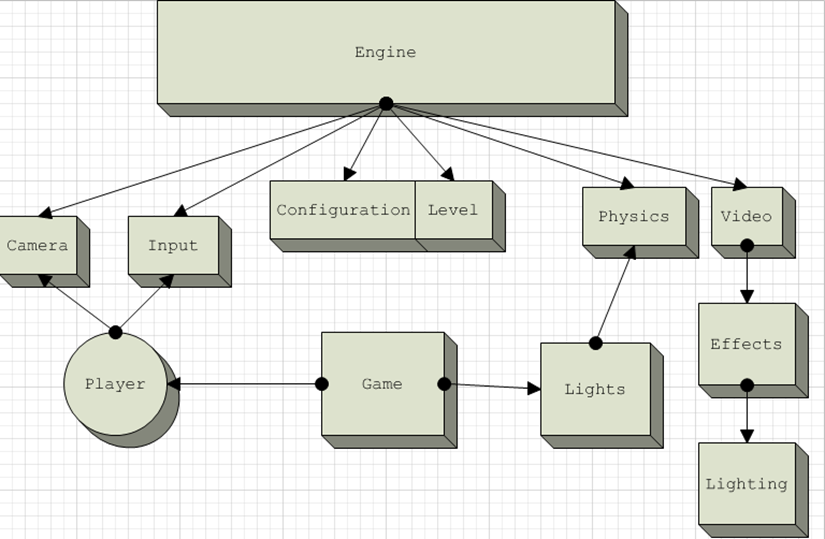
### Gameplay

The gameplay of Ne+ is associated heavily with the visuals by the lighting elements. When the player object or other designated game objects are touched by light objects, the lights implement their effects on the game objects. This describes the core element of the gameplay. Anti-gravity lights make game objects float; friction lights change the friction coefficients of platforms; one light projects a light platform; a null light nulls the effects of all other lights; all of these lights can be combined to create different and intriguing effects dynamically as the game is played. The light objects utilize engine elements to implement their effects; this can be viewed in the block diagram in Figure 3 where the Light component utilizes the Physics component of the engine to implement effects – this is how the anti-gravity light manipulates other game objects through the physics simulation.

## Block Diagram

In designing Ne+ at a high level, one of my goals was to have some sort of engine class that would hold much of the global sort of video game resources that I thought almost all game code would need to be able to access. This is clearly represented in the block diagram below.

Figure 3: Block Diagram



The engine is basically a concept for the elements below it to fit into. The elements it actually contains, like the Camera, Input, Configuration, Level, Physics, and Video elements are some of the crucial pieces of any video game that I thought should be easily accessible. The Game component, being the other main element in this block diagram, represents the game-specific functionality that details the particular pieces of Ne+ that make it different from a standard game, and as such are organized separately from the engine. The player and lights are the most important pieces of it, the lights being the core functionality of Ne+. The lights are a good example of a game element that utilizes the engine, shown in the diagram to be using the physics component to manipulate environment. Video effects like lighting are also clearly represented as part of the engine, being a generic need that the game component can further customize to its specific needs on its own.

## Summary

The functional description of Ne+ shows a core collection of functionality that other game components can access, query for information, or take advantage of other functionality or effects that the engine provides. This creates a solid and organized collection of game pieces that can be utilized or put together by the Game to create the game-specific functionality that Ne+ requires.

# Chapter 4 Detailed Description

## Overview

The detailed description of Ne+ entails how exactly the major architectural components of the project work together, and then how some of the components themselves work. In the Architecture section, the major partitioning of the different components in the project is described. In the Detailed Design section, many of the underlying components in the architecture are explored.

## Architecture

The architecture of Ne+ was decided upon after a lot of careful research exploring existing game technologies and engines that were open-source. Part of the driving concept behind the architecture was to not re-write anything that was already publically available, in an effort to save as much time during development as possible. A look into existing XNA game engines did not reveal any hopeful prospects, which left Ne+ collecting as many different open-source components as possible and integrating them into a custom engine.

The initial architecture concept is located in Figure 3, wherein a superclass called the Engine oversees the third-party technologies I have chosen to use with Ne+ as well as any of the major pieces of the project that I developed myself. A list of the architectural components can be found below:

* Engine (loads and updates all components and major pieces of the project)
  + Camera (provides the player viewpoint into the application)
  + Configuration (loads configuration values from a text file and exposes them at runtime)
  + Input (this component offers player input for reading)
  + Level (if a level has been loaded, this piece of the engine exposes its public components)
  + Lighting (this component manages the lighting simulation)
  + Physics (this component manages the physics simulation)
  + Video (this component manages video information)

It becomes clear through noting the different architectural components that there are a few very key components, namely the Level, Lighting, and Physics components. Between these three, the majority of the system communicates. The level component uses the lighting and physics components to instantiate the majority of the game, albeit indirectly with a host of components I wrote to abstract concepts into more usable pieces of the game and gameplay.

## Detailed Design

### Camera

The camera component in Ne+ takes care of a few different responsibilities. First and foremost, the camera provides a matrix which is used in every single call to XNA’s SpriteBatch.Draw() method, which provides the ability to zoom in and out, and translate the camera in whatever direction need be. The camera also provides a variable for an object to follow, which in turn allows the player or any other object to be followed. It also can return a rectangle representing the area of the game world currently visible, so that it can be known which objects actually need to be drawn when it comes time to render the current game scene.

### Level

The Level component of Ne+ is where much of the initial magic happens; when a level is loaded, there are many things happening. Based on a Tiled map, artwork, collision information, and lighting information, as well as player spawn and enemy spawn information, is all read and followed accordingly. Collision objects are created with the Physics component, so that the physics simulation matches the visuals on the screen. Light objects are created within the level with help from the lighting simulation component, which is discussed in more depth under the Lighting detailed design section.

### Lighting

The Lighting component is responsible for drawing a light set over the game artwork. Certain game objects are then drawn on top of the lighting if they are supposed to disregard the lighting entirely. This makes the order in which this component is drawn crucially important, but as it inherits from a base Component object it has a draw order that can be set easily. Lights are added and removed from the simulation, and manipulated accordingly.

### Physics

The Physics component of the game is arguably the most important, guiding the movement and collisions of all game objects. It also is responsible for sensing when objects are under lights, which is the core gameplay concept in Ne+. The Physics component uses the Farseer Physics Engine, which makes the process of creating and manipulating objects in its physics simulation a quick and easy one.

## Summary

The architecture of Ne+ is very clearly modular, separating the large pieces one imagines being part of a game engine into their own components, many of which like the Lighting and Physics components representing actual third-party libraries meant to solve such pieces of a game. This makes development simplified and as easy as possible, allowing precise knowledge of where exactly in the game codebase particular functionality lies.