

2/10/2006

Team #7: Pez

Project: Empty Clip

Members: Alan Witkowski, Steve Huff, Thos Swallow, Travis Cooper

Document: SDS

1. Introduction

1.1 Purpose of this document

The purpose of this document is to outline our design decisions. We will describe our system architecture and individual components within our architecture. We will discuss relations to other products and talk about trade-offs and solutions.

1.2 Scope of the development project

Empty Clip is an Action RPG whose purpose is to entertain. The game allows upgrading and customization of characters and weapons. Mind-boggling puzzles will be key to keep the gamer entertained and engaged. Multi-player networking will add collaboration and teamwork.

1.3 Definitions, acronyms, and abbreviations

2D – Two dimensional

3D – Three dimensional

GUI – Graphical User Interface

1.4 References

Davis, T., Jackie, N., Shreiner, D., and Woo, M. *OpenGL Programming Guide*, Pearson Education, Boston, MA, 2004

NeHe Productions. "NeHe Productions: OpenGL Lessons", 8 December 2001. Online. Internet [25 January 2006]. Available WWW: <http://nehe.gamedev.net/>

Norvig, Peter and Stuart J. Russell. *Artificial Intelligence*, Pearson Education, Inc., Upper Saddle River, NJ, 2003

1.5 Overview of document

In section 2 we will describe various components and their relations to our game.

In section 3 we will give a detailed description of our components.

In section 4 we will describe outside products and code reuse.

In section 5 we discuss design decisions and trade-offs.

In section 6 we add pseudo code for our components.

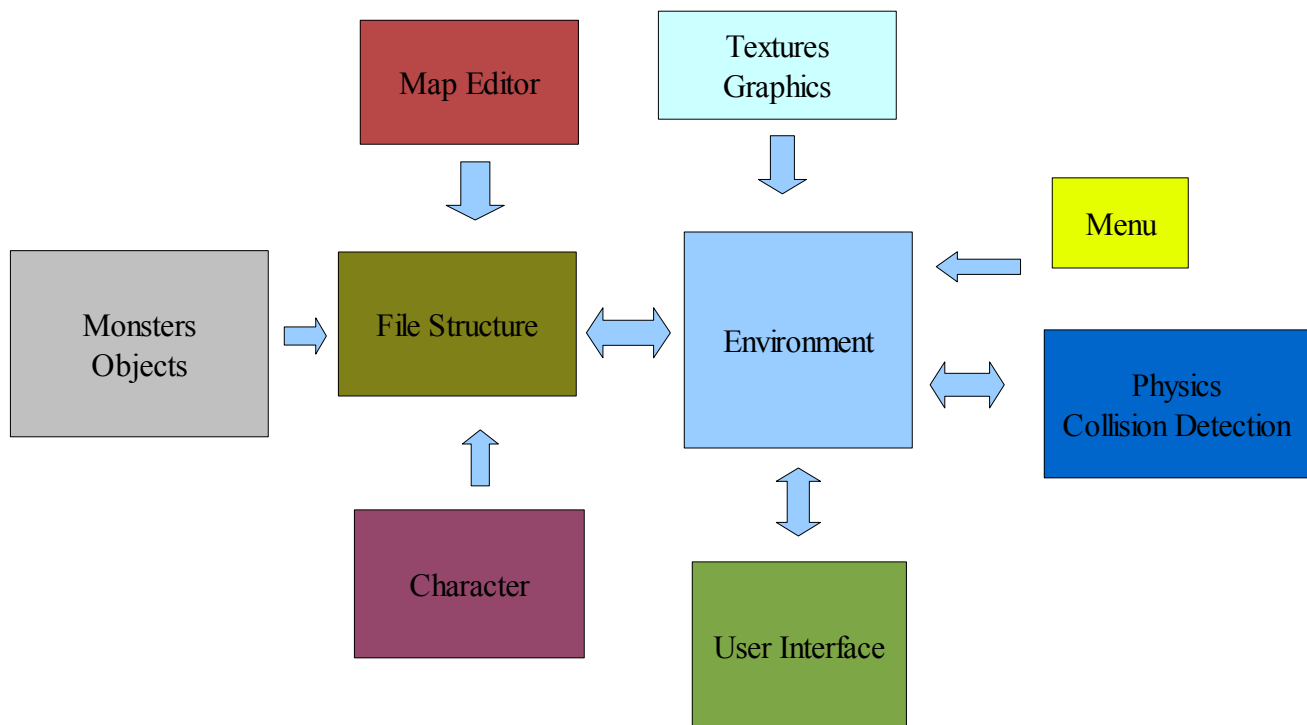
In section 7 we will build an appendix for our document.

2. System architecture description

2.1 Overview of modules/components

- Game Environment: Playing area for character. Includes monsters, objects, textures, boundaries, etc.
- User interface: Information about the current state of the character and game, including inventory, health, and skills
- Graphics: OpenGL system used to control and maintain what the user sees on the screen.
- Character: An object in the game that is controlled by the user, using keyboard and mouse.
- Monsters: animated objects on the screen that interact with the character.
- Objects: unanimated textures(objects) that the character can interact with.
- Textures: the smallest unit of the game playing area. The playing area is set up in blocks.
- Physics/Collision Detection: The system used to ensure that the character interacts with the environment in a logical manner (e.g. Not walking through walls).
- Menu: The initial setup of the character as specified by the user.
- Map Editor: The tool used to create environments in the game.
- File Structure: The system used to keep track of object, characters, and monsters, as well as the map editor.

2.2 Structure and relationships



2.3 User interface issues

GUI – The gamer will need to be able to navigate the in-game menu system with ease. All game configurations, including volume and keyboard mappings, should be easily accessible. The interface should be easy to read and intuitive.

Game – The gamer will navigate his/her player with the keyboard and mouse. The mousing will control the aiming direction, and clicking will attack with the weapon. The mouse will also control the interface. The options menu will always be available by hitting the escape key.

3. Detailed description of components

This section is the main focus in version 2.0 of the SDS, the detailed design

4.0 Reuse and relationships to other products

We are using OpenGL for our graphics engine, which provides us with a nice interface to hardware accelerated graphics. We are also using SDL for our keyboard and mouse input, networking engine, and audio system. For drawing fonts, we are using the GLFT_font library. We are reusing collision detection and ray casting code from Alan's previous projects.

Due to using these open source libraries are subject to limitations and standards that were set by the designers. Where possible, we have purged excess code from these libraries.

One resource that we are not reusing is a particle engine. The reason is because most particle engines provide more functionality than we need.

5.0 Design decisions and trade-offs

One major design decision was to make the game 2D instead of totally 3D. This reduced the complexity of animation, level design, collision detection, and testing. However, the main trade-off is realism in character animation and level design. Another decision was to make the game tile based. This makes collision detection and the data structures more simple. One drawback to using a tile based engine is that it makes it hard to make levels that don't look tile based.

6.0 Pseudo code for components

Section not applicable in version 1.0

7.0 Appendices

Section not applicable in version 1.0