Accent
Game Proposal
By: Wayne Jones, Leonard Law
CS 3113

Table of Contents

Introduction

Accent is a 2D shoot-em-up that adds rhythmic elements into the core gameplay. Fans of shooters and rhythm games alike will find this game challenging, as a great deal of concentration is required to excel in this game.

Description

The objective of the game is simple. Players have to obliterate all enemy units in as few rounds as possible. In most cases, eliminating the main enemy "boss" will suffice. In any case, all units may assume three basic phases: attacking, evading, and maintenance. Both sides cycle through these phases in turn.

The attacking phase will involve dealing direct damage to enemy unit(s). This is done via directly firing projectiles, or having a slave unit deal the damage in proxy. The evasion phase consists of moving the unit so that enemy projectiles do not hit the unit. Depending on the current game environment, this may be anywhere between trivial or impossible. In an attempt to balance the possible difficulty curve, the third mechanic is introduced. The maintenance phase is where units can recover or strengthen themselves. This is done by pressing keys in accordance to a rhythm based on the track is currently playing. Basic rhythm patterns will grant a small bonus, while more complex patterns will grant larger bonuses, and/or a special attack or skill that may be used in the next active phase.

A battle starts from an initial setup phase. In this phase, both players assume the Maintenance phase and press keys according to the rhythm. After a set amount of time, the battle cycle begins. The first player assumes the attacking phase, while the second player assumes the evasion phase. After the first player's attacking phase expires, the roles are reversed. Next, both players will again enter the maintenance phase. This cycle repeats until either or both players have no energy when entering the maintenance phase. When both players' tracks have expired and neither player is out of energy, players may choose different tracks and battle again, or enter sudden death, where an impossibly hard track is chosen and gradually sped up until a player cannot keep up, where the player that remains is declared the winner.

Key Features (Programming)

The game engine will consist of two parts. The rhythm engine will be responsible for determining whether or not the rhythmic aspects of the game were properly played. The shooter engine will be where the actual shooting and projectiles will be handled.

The player will have control over the unit's position at all times during a battle. While attacking, players only have control over where the unit will fire at, indirectly affecting the unit's orientation. All slave units may or may not use this point as a reference target when attacking. While evading, players lose the ability

to control where to fire at. Instead, players are able to directly affect their unit's orientation. This allows for finer control over evasive maneuvers as needed.

The player will control the unit's position using the keyboard as input. The keyboard will be used for positioning as well as shooting. Mouse input will be used for special occasions and can only be utilized, for instance, when a player gets a max combo on an entire maintenance phase.

Because there is a lot going on in this game, the user interface has to accommodate for this complexity. The interface will utilize tweens to cleanly switch between the various player phases. This will be done by utilizing opacity fades, camera shifts, as well as sprite transformations to emulate 3D effects where they may be needed. If possible, a particle engine will also be integrated into the graphics engine to simplify rendering of various particles, such as energy streams, explosions, light beams, etc.

Generic input interfaces will be integrated into the game engine. This is to allow for dynamic switching of input devices, and easier integration of networked battles. The typical user will only have a keyboard and a mouse as input, which are unsuitable for serving more than one player. A working goal for the project is to have single-system multiplayer via the use of USB controllers. If time allows, there will also be support for 2-player multiplayer gameplay via TCP/IP. We will have to find a way to sync the two players' clients together, and account for latency

between the players.

An AI will be designed to control (potentially) both players. Because there are only two players, the AI can be implemented using A* or a variation of it as the memory overhead would not be too great. This AI will have a scalable difficulty, and may be adjustable mid-battle.

Genre

This is a sci-fi shooter rhythm game. Player attention will be evenly divided between keeping a steady rhythm and controlling their onscreen units. Since there are not a lot of games that fall under this genre, Accent will be a great addition to the sci-fi shooter rhythm genre.

Platform/Target Audience

The game platform that will be used is XNA that is coded in C#. The game is made for the PC but XNA can be ported for the Xbox 360. This game is targeted at PC gamers, though it can be easily made to target console. As a PC game, it performs well on single-player, as well as online multiplayer. Given the complexity in player controls, it will be poor in handling multiplayer using a single input device, namely a single keyboard and mouse. However, due to development time constraints, the PC would be the ideal choice in developing a game in as fast a time as possible.

Mid-Project Objective

By the middle of the project, we plan to have the game playable at its most basic stage. Instantaneous attacks such as bullets and projectiles will be complete. Damage over time (beams) and/or periodic attacks (slave drones) may or may not be present, but not fully functional.

The rhythmic portion will have single note recognition complete. Held notes may be playable at this stage, but extremely unforgiving. Other notes are not planned, but may be added as development is started.

Other aspects of the game will most likely appear to be rather primitive. The overall interface for mode selection and track selection has not been planned, so at this stage it will be barely usable to allow demonstration of the main aspects of the game. Basic sound effects will be implemented by this stage.