XNA Code Camp 2010

Awesome Jungle Project Report

Group 4
Li Aoke
Xing Jifeng
Antti Knutas

1. Introduction

The Awesome Jungle is The Jungle Game coded with the latest Microsoft Game Studio tools that allows one to develop simultaneously both for any major Microsoft platform. This allowed the game to be developed simultaneously both for the Windows platform and the Xbox 360 game console. The game itself is an old Chinese strategy game that is still commonly sold in boardgame form. The computer game is very similar to the original, but with rules checking, animation, special effects and other features added.

Newcomers might find the game easier to start because the game also checks for valid moves and prevents the player from making errors. Of course one still has to still learn and develop the strategy to beat the opponent or the machine AI.

1.1 Game Rules

The goal of the game is either to move a piece onto a special square, the lair, on the opponent's side of the board, or capture all of the opponent's pieces. Each side has 8 pieces representing different animals, each with a different rank.

Higher ranking pieces can capture all pieces of identical or weaker ranking with the exceptions: The Mouse (lowest rank) may capture the Elephant (highest rank). Each piece moves one square horizontally or vertically (not diagonally). Rat, Tiger and Lion have some movement special ability related to the water squares. The player may capture any enemy piece in one of the player's trap squares regardless of rank.

1.2 Use Cases

The use cases related to the machine are very common to what one might expect from a board game. There is only one user group, too, which is the player. The following list has the most common cases that might occur within the game:

- Start new game
- End game
- Make move
- Win game
- Lose game

2. Technical implementation

The main core of the game is in the shape of a loop that tries to execute as fast as possible, doing input checking, graphics redrawing and other logic functionality with each cycle. This is in sharp contrast with other listener-based development methods, but very common to games development. The game itself was coded in C# with utilizing the XNA game libraries that allow easy playback of audio and drawing graphics on the screen.

2.1 Graphical Interface

As chess like broad game, gorgeous graphic effect and exciting sound effect are not necessary. But to fit for the game background, animals fight in a jungle, we design to add some basic fight effect to increase the playability of the whole game.

The sound effect of animal fight includes a punch, a slap and a wail which in fact come from a woman to pretend like an animal. Use SoundEffectInstance.play() command can easily play the sound when graphic effect happen. The graphic effect contents smoke

and sparks. The material shows in figure 1.1.

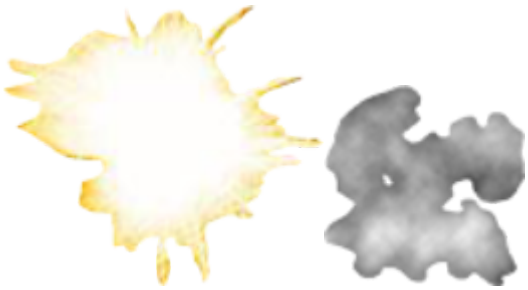

Figure 1.1 the material of spark and smoke

Several of the sparks and smoke will explosive from the start point which the node two animals fight. Figure 1.2 shows a screenshot of the effect.

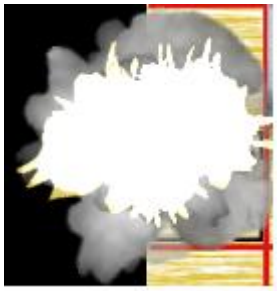

Figure 1.2 screenshot of the fight effect

Each spark and piece of smoke has been set as a particle of the whole effect. Every time select a random number of them move from the start point in random direction and then disappear. Each particle has its own initial speed, acceleration, lifetime and rotation speed. All those parameter have been randomly selected from a defined set. So every time the effect have same style but different which make it more reality. After those spark and smoke disappear, the winning animal will stand on its purpose space. The loser of the fight will out of the game, which means will not been draw on the screen again. That process have been control by game logic.

2.1.1 Multiple screen and menu system

The game needs menu system. At beginning when player enter the game, should show him the main menu, which give selection for single player (player with AI) or multiple player. During the game, should let the player pause the game then continue it or just quit the game.

To implement that, need multiple screen. For each menu, like main menu and pause menu, give it's own screen. Two screen call game play screen and multiple play screen have all game logic and game graphic. The game is only running on those two screens.

The game play screen contents AI logic and multiple play screen contents multiple player control logic. For pause menu screen and quit confirm screen, the background is semi opaque. The effect of that show in figure 2.1. The game broad show through the semi opaque black background so that player can save their game step during pause and continue when resume .

Figure 2.1 semi opaque background paused menu effect

There is a screen manager to manage all those screens. The screen manager have the only spritebetch of whole game system. Everything is drawing on that. Each time base on the player input, screen manager determines which screens should be drawn on the spritebetch in some order. For instance, when player press Esc key to call pause menu, screen manager should keep drawing the game player screen and draw pause screen above that. Let the player only control the pause screen. Because all update relate to the player input, the game screen will keep it's own state so that player can continue the game with no problem.

2.3 Board Map and the AI

Object-oriented programming techniques were used to represent the layout of the board inside the game: Each node, or square, on the board is represented by its own object which was linked to the neighbouring nodes. Each node has functions for manipulating the contents, and variables for containing the game piece objects. This helped manipulation of the board by the rules structures and the AI.

The AI uses basic route-seeking algorithms and preprogrammed action-reaction responses. In addition to trying to reach the goal with some piece, it checks if it is threatened or if it can make an attack. It is simplistic in a way that in can be still tricked easily if one knows the programmed in rules, and it does not try to forecast the game further than one single turn. This would be a definite point of development.

4. Development Methods

One of the key factors that made this project successful was well defined work distribution. This means that every member of the team had his own area to focus at. Antti focused mainly at logical structure of the game, the board and the AI, Aoke did most of the effects and menu coding and Jifeng worked on codifying the rules and making the graphical presentation of the board. Of course, these responsibilities were overlapped, especially during the dead line-style coding phase in the end of the code camp.

As a primary development tools, Microsoft Visual Studio 2008 and the XNA Game Studio extension were used. This time SVN code repository technology was not used, which caused some problems for development because sometimes different versions or the branches of the code on different machines diverged. Compared to earlier projects where SVN was used, the integration took considerable effort. The development process involved also considerably discussion and interaction among the team members. This also promoted the XP (Extreme Programming) -style development. The code is a bit clumsy at some places. This is because of very limited time and the need for getting out functional pieces of code very fast. The style of coding is similar to XP: the test code were often written before the actual code to ensure the functionality. All the testing with the development machines, which was very easy because of the way the game studio allows to choose the platform to compile the binaries for.

For documentation tasks of the code camp, Google Documents. was used for the documentation, presentation and report writing and the course wiki was used for goal tracking. The team was not able to work all the time in one centralized place so some remote work had to be done and these tools are very suitable for that purpose. The amount of project planning was rather small because of restricted time. For its part, this caused some problems that could have been prevented by considering some things beforehand. The team realizes the importance of planning, but it is self-evident that in this kind of course, the focus is on coding. However, the used methods and the tools were suitable for this project and the result was very successful.

5. Summary

There were a great amount of code camp spirit among the software team. The used tools and techniques were very suitable for the project but all the members also learned a lot and identify those things that could be done better in future projects. The software team reached the goal that was set during the first day of the code camp and the result of this is a fully functional Xbox360 application.