HPX design document

Aquila

November 11, 2001

1 Introduction

HPX is an evolutionary computer game development project. Starting off as an extremely simple game, the idea is to slowly evolve towards the ultimate goal which is a relatively complex and fun game. HPX refers to the development project, not the game itself. The name of the developed games will be decided during implementation.

2 Goals

The goals of this project are as follows:

Practice. To refine and improve the programming and software development skills of the developer(s).

Show off. To show that the developer(s) are capable of writing quality software and possibly improve employment possibilities. To actually finish a project which the developer(s) can be proud of.

Enjoyment. To actually do something which all young hackers dream of doing. To be able to play an innovative and exciting game instead of whats on the market right now.

Waste time. To constructively use up time which would otherwise be spent *playing* games.

3 Approach

3.1 Development process

The project will be completed in n phases, where $phase_n$ is the implementation of the ideal game design. The value of n will be undetermined until the completion of $phase_{n-1}$. For a $phase_i$, a new section will be added to this design document specifying the new features or modifications to $phase_{i-1}$. $phase_i$ will then be implemented according to the specification. After implementation, the game would be released for testing and evaluation, which will be documented, before moving on to $phase_{i+1}$. During evaluation, the ideal design may also be modified, if necessary.

3.2 Requirements

Each phase will meet the following requirements.

Cross-platform support. It must support for both Linux and Windows. Possibly others in the future.

Freely distributed. It must be distributed along with the source code without charge.

3.3 Implementation notes

Some of the implementation details are set and will remain the same for all phases.

Development will use the Simple DirectMedia Layer (SDL) library[1] which has been ported across multiple platforms and will make porting each phase an easy task. OpenGL[2] may be used in later phases if necessary.

The programming language used will be either C or C++.

All of the code will be released under the GNU General Public Licence[3] (GPL).

4 Ideal design

4.1 Overall

This is the vague description of the current ideal design.

A team battle game, where one (or more?) players act as the "general" overseeing the battle and resources, and issues orders (real-time strategy style) to units, while other players act as "sergeants" who play action style and carry out those orders or perhaps lead a small company into combat. Have a nice storyline and even have campaigns which one team can play vs AI or other teams.

4.2 Game world

The game world consists of a map of the terrain, a set of buildings, and a set of objects. Buildings are unmovable obstacles. Some buildings can be entered. Objects are the super class of players, projectiles, mines, turrets, and so on. Each subclass have their own "methods".

4.3 Gameplay

Currently the ideal gameplay is similar to Tribes 2, except from a set, isometric perspective.

4.4 Graphics

The objects would be rendered 3D but from a set perspective. The terrain would be pre-rendered.

[1] http://www.libsdl.org
[2] http://www.opengl.org
[3] http://www.gnu.org/copyleft/gpl.html

5 Constraints

There are some foreseeable problems which will need to be overcome eventually at a certain phase.

Lack of manpower and talent. There is currently only one developer, who has limited abilities. The developer has some programming skills but very little skills in graphics, sound, music, and plot writing. More people will need to contribute to make this project viable.

6 Phase 1

6.1 Aim

The aim of this phase is as follows:

- Start the project off with a very simple, but working, game.
- Get started on a simple graphics, networking, and game-world engine.

6.2 Specification

6.2.1 Graphics

Isometric view. The viewpoint is a constant angle down onto the players.

Sprite-based. Players and projectiles are a set of sprite images. Purple pixels (`0x010001`...`0x800080`...`0xff00ff`) designate the team colour. There is one sprite for each of the 16 directions.

Shadows. Each sprite has a 1-bit shadow sprite, which is the shape of the shadow if an area light was shone directly above the object (e.g. a circle for a spherical object.)

No animation. Sprites are not animated.

No terrain. The background is a plain colour.

6.2.2 Game world

Two players. The players are the only objects in the game world. There will be only two players in each game.

Tanks. The players are small tank units with a turret which rotates independently of the body.

Projectiles. The players fire projectiles at each other. There may be any number of them in the world at a certain instant.

Placement. Objects have (x, y, z) co-ordinates. The (x, y) co-ordinates are used to determine the center of the object shadow. The z co-ordinate is used to determine where to draw the sprite. An object is drawn in front of another if the y co-ordinate is greater.

Collision detection. Two objects (player, boundary, or projectile) are considered to have collided if their shadow sprites touch each other, or if the shadow sprite exceeds the boundary.

6.2.3 Gameplay

Controls. The players target using the mouse (and rotates the turret), and moves using the keyboard. The turret rotates to face the mouse pointer and the tank body rotates left, right and moves forward, backward using the left, right, up, down arrow keys respectively.

Firing. A tank fires bullets with a cooldown rate. The speed of the bullet is determined by how long the mouse button is held. A bullet travels in a straight line and disintegrates a set time after firing.

Health and damage. A tank has a certain number of health points. Collision between players destroys both players. Collision with a bullet destroys the bullet and reduces the health points by a set amount. When a player's health reaches zero, the game is over.

6.2.4 Networking

Server-client. One player starts the game, which starts a server and a client connecting to it locally. Another player starts just the client and connects to it via a host name and port number. Game states are broadcast from clients to server and vice versa via a TCP connection.

7 Notes