|  |
| --- |
|  |
| Time Warrior |
| Game Demo Document |
|  |
|  |
| **6/14/2011** |

Alex Beerman – Lead Designer/Art Design/World Designer

John Parks – Lead Programmer/Artist/Combat Designer

Joe Ortega – Experience Designer

Matthew Morrill – Project Manager/Co-designer

|  |
| --- |
| This is the Game Demo Document of the game Time Warrior. It contains portions of the Game Design Document, Technical Design Document, and some code samples. |

Table of Contents

[Game Design Document 4](#_Toc295829749)

[Overview 4](#_Toc295829750)

[Premise 4](#_Toc295829751)

[Introduction 4](#_Toc295829752)

[Genre 4](#_Toc295829753)

[Target Demographic 4](#_Toc295829754)

[Technical Specifications 4](#_Toc295829755)

[Game Mechanics 4](#_Toc295829756)

[Core Functionality 4](#_Toc295829757)

[Game Play 5](#_Toc295829758)

[Game Elements 8](#_Toc295829759)

[Technical Design Document 11](#_Toc295829760)

[Executive Summary 11](#_Toc295829761)

[Equipment & Tools 11](#_Toc295829762)

[Engine Evaluation 11](#_Toc295829778)

[Platform Evaluation 11](#_Toc295829780)

[Coding Structure 12](#_Toc295829781)

[Story of Our Problems 14](#_Toc295829782)

# Game Design Document

## Overview

### Premise

Travel through the ages to gain redemption, and "freedom". Fight, explore, loot, and level as you figure out the history of the world you live in.

### Introduction

You are a prisoner for an unknown crime. Your freedom can be gained by going through the ages and liberate the demons infecting time. You are armed with a gauntlet that can transfer energy from the universe and creatures into “magic”. You’re goal is to get to your freedom.

### Genre

Action Role-Playing Game

### Target Demographic

T for Teen

### Technical Specifications

Processor: Intel Core 2 Duo 2.2 GHz

RAM: 2 GB

OS: Windows XP SP 3 or Windows 7

Hard Drive: 115 MB

Video Card: Intel 965 Chipset Mobile

#### Platforms

PC

#### File Formats

.txt, .tbmp, .bmp

#### API

SDL

#### Code Language

C++

## Game Mechanics

The main mechanics of the game include the gauntlet, which creates the magic and weapons. The gauntlet has memory storage, which acts as the inventory system, which allows the player to switch their magic, weapons, and armor on the fly, using chips that contain the data.

### Core Functionality

#### Data Files

All data files are stored in .bmp’s, or .txt’s.

#### Resource Loading and Unloading

Resource Loading and Unloading is done as the game runs, loading from .txt files and .bmp files. Saving to files will require another .txt file.

### Game Play

The game play uses a top-down camera for ease of use and navigation. Since there would be no reason in which to rotate the camera, there doesn’t need to be a camera rotation. The controls of the game would be a keyboard and mouse. The default keys are as shown:

|  |  |
| --- | --- |
| Default Key | What it would be used for: |
| W | Move up |
| A | Move left |
| S | Move down |
| D | Move right |
| Left Click | Activate whatever attack (magic or weapon) is configured as 1st attack |
| Right Click | Activate whatever attack (magic or weapon) is configured as 2nd attack |
| I | Open up the Inventory |
| H | Use Health Potion |
| E | Use Energy potion |
| 1-3, 5-7 | Quickly change left-click attack (1-3) or right-click attack (5-7) to another attack of the same type (i.e. change Basic Fire to Advanced or Expert Fire). |
| Space Bar | Activate an NPC’s Dialogue when in range |

HUD design:



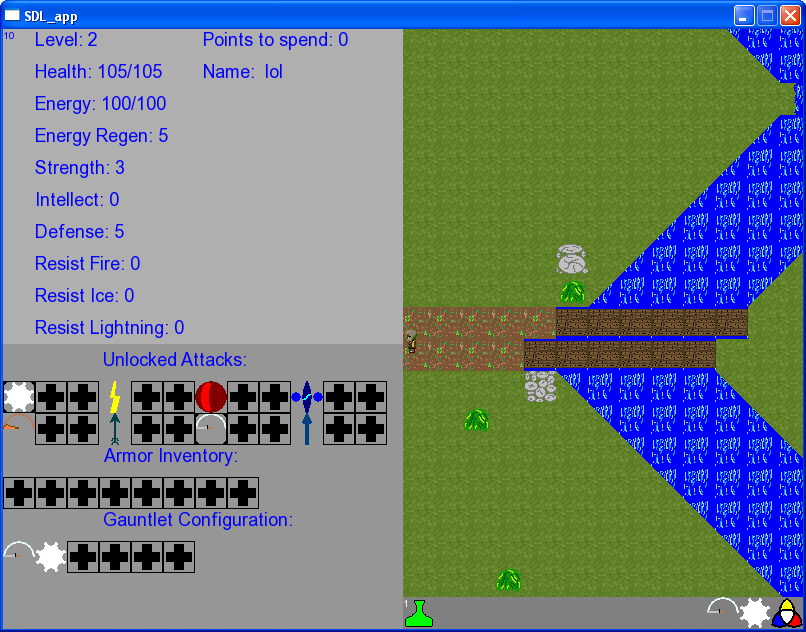
* Health, energy, and experience bars in lower-left corner
* Stat button, in lower-right corner, opens a menu to show player’s stats when clicked
* Icons for attacks set as ATK1 and ATK2 in lower-right corner, near stat button
* Potion Icons for Health (Red) and Energy (Green)

Experience System:

The player’s experience requirement for each level is calculated as:

Experience requirement = “current level” + “experience requirement from previous level”

|  |  |
| --- | --- |
| Level | XP Needed |
| 1 | 2 |
| 2 | 4 |
| 3 | 7 |
| 4 | 11 |
| 5 | 16 |



Upon leveling up, the player can choose to boost either their Strength or Intellect, which affects their other stats:

|  |  |  |
| --- | --- | --- |
| Stat | Description | Relation to Other Stats |
| Health | Life points | Increases with Strength |
| Energy | Used to handle cost of attacking | Increases with Intellect |
| Strength | Increases damage with Weapons | Increases Health and Defense |
| Intellect | Increases damage with Magic | Increases Energy and Resistances |
| Defense | Used in combat with Weapons | Increases with Strength |
| Fire Resistance | Used in combat with Fire Magic | Increases with Intellect |
| Ice Resistance | Used in combat with Ice Magic | Increases with Intellect |
| Lightning Resistance | Used in combat with Lightning Magic | Increases with Intellect |

#### Game Initialization

Initializes with only one player, at which point the Hub world loads. No monsters are present in the Hub World, but monsters are loaded dynamically once the player steps into the Portal.

### Game Elements

Magic is categorized by its element, which can be Divine, Fire, Ice, or Lightning. Each element has 3 levels of spells: Basic, Advanced, and Expert. When applied to the same target, spells can combine to create combinations.

#### RPG Elements

An Inventory system will be implemented that WILL NOT pause the game. Players can switch whatever chips they currently have equipped with another chip whenever they want. However, they can still be attacked, and can still move. Each enemy defeated gives a certain amount of character exp, and each use of a chip gives a small amount of Chip exp. Each character level gives one chip level for the player to use as they wish, which helps them specialize in a certain element or weapon type. Extra chips, which can be found after the enemies are killed on the ground, can be sold for a certain amount of chip exp, depending on the chip sold, which can be applied to any chip chosen by the player.

#### Enemies

Each enemy gives the player a certain amount of experience, based on their relative levels:

Total experienced earned = “enemy level” \* (“enemy level” / “player level”)

All enemies are strong and weak against a certain type of magic, depending on what they are categorized as (made of, covered in, or attributed as such):

|  |  |  |  |
| --- | --- | --- | --- |
| Material | Examples | Strong Against | Weak Against |
| Wood | flora, plants, vegetation | Ice | Fire |
| Fire | fire, magma, lava | Lightning | Ice |
| Earth | Ground, dirt, mud, sand, rock | Lightning | Fire |
| Metal | metal, gold, silver | Ice | Lightning |
| Water | Water, ice, water-vapor, steam | Fire | Lightning |
| Air | Gases, wind, lightning | Ice | Fire |
| Rubber | Rubber, elastic | Lightning | Ice |
| Light | Light, heavenly, angelic, divine | Fire, Divine | Ice |
| Dark | Dark, undead, shadows | Ice | Fire, Divine |

#### Weapons

There are four main weapon types, Slash-Pierce-Blunt-Range. Each weapon type also has 3 levels to it, getting more destructive as they go. Each weapon changes with the age dependent on what they are. For example a crossbow in the medieval era will change to a gun in the modern era. More information can be found in the Distinctive Player Characteristics section.

Total damage done with weapons is related to the player’s strength, increasing the weapon’s base damage by a certain percentage:

Total weapon damage = “base damage” \* ((“base damage” + strength) / “base damage”)

When hit by weapons, total damage taken is decreased by a certain percentage using the defense stat:

Total damage taken = “total weapon dmg” \* (“total weapon dmg”/ (“total weapon dmg” + defense))

Each weapon type has a distinctive use or effect:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Blunt | Range | Slash | Pierce |
| Effect | Knockback | Projectile | High damage | Chance to inflict extra damage |

Each category follows the pattern described above.

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Basic | Advanced | Expert |
| Blunt | Blunt Bash | X-Bash | Hurricane Bash |
| Range | Ranged Shot | Double Shot | Tri-Shot |
| Slash | Slash Strike | X- Strike | Hurricane Strike |
| Pierce | Pierce Stab | Running Stab | Stabby-Stabby |

#### Magic

Like weapons, total damage dealt with magic is related to one of the player’s stats. The player’s intellect is used to increase base magic damage by a certain percentage:

Total magic damage = “base damage” \* ((“base damage” + intellect) / “base damage”)

When hit by magic, total damage taken is decreased by a certain percentage using the appropriate resistance, which accounts for fire, ice, or lightning damage:

Total damage taken = “total magic dmg” \* (“total magic dmg”/ (“total magic dmg” + resistance))

Using the above formula, divine damage is calculated by assigning a certain value to “resistance” based on the attributed material:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Light | Dark | Everything Else |
| Divine Resistance | 1/2 “total magic dmg” | 0 | “total magic dmg” |

Each level of spell acts differently in terms of how it operates:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Basic | Advanced | Expert |
| Targets to Hit | Projectile | Self-centered area-of-effect | Click-centered area-of-effect |

Each magic type has a distinctive use or effect:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Divine | Lightning | Fire | Ice |
| Effect | Heals allies | Chance to inflict extra damage | High damage | Damage over time |

Each element follows the pattern described above.

|  |  |  |  |
| --- | --- | --- | --- |
| Element | Basic | Advanced | Expert |
| Divine | Divine Light | Divine Barrier | Sanctuary |
| Lightning | Lightning Bolt | Chain Lightning | Thunderstorm |
| Fire | Fireball | Fire Blast | Armageddon |
| Ice | Ice Shard | Ice Frost | Blizzard |

# Technical Design Document

## Executive Summary

This is the Technical Design Document that contains all the possible tools, evaluations, assets, and overall development plan to make the game Time Warrior.

## Equipment & Tools

Main equipment will be Visual Studio 2008 C++. A Map Editor tool will be created during development to assist in creating the maps. Map system will use characters for each individual Tile. A map file looks like this:

HBBBBBBBBBBBBBBBBBBBBBBBBBD

MGGGGGGGGGGGGGGGGGGGGGGGGGV

MGGGGPGGGGGGGGGGGGGGGGGGGGV

MGdGGGGGGGGGGGGGGGGGGGGGGGV

MGGGGGGGGGGGGGGGGGGGGGGGGGV

MGGGGGGGGGGGGGGGGGGGGGGGGGV

MGGGGGGGGGGGGGGGGGGGGGGGGGV

MGGGGGGGGGGGGGGGGGGGGGGGGGV

MGGGGGGGGGGGGGGGGGGGGGGGGGV

MGGGGGGGGGGGGGGGGGGGGGGGGGV

MGGGGGGGGGGGGGGGGGGGGGGGGGV

MGGGGGGGGGGGGGGGGGGGGGGGGGV

HBBBBBBBBBBBBBBBBBBBBBBBBBD

Source Control will be using Tortoise SVN + Google Code.

## Engine Evaluation

The engine will be state-driven, consisting mostly of states such as “Title Screen” state or “Game” states.

## Platform Evaluation

Time Warrior will not be multi-platform, being made specifically for the PC. Minimum specs are:

OS: Windows XP SP 3

Processor: Intel Core 2 Duo 2.2 GHz

Video Card: Intel 965 Mobile Chipset

RAM: 2 GB

Main build process will be done on Windows XP SP 3, and Windows 7, all using Visual Studio 2008.

## Coding Structure

Coding structure will be mostly hierarchical, using inheritance from a base class. World will be the main holder of information, and each Entity needs to know some basic functions from World in order to determine collision. Each Entity will be aware of its own surroundings, and be in charge of its own collision.

Inheritance Diagram (Entity is base class)

World Composition

|  |
| --- |
| World |
| //divide the world into grids of entities  TemplateVector<Grid> m\_mapOfEntities  //the tiles it takes to visually make the world  TemplateVector<Tile> m\_mapOfWorld; |
| //various functions to create world from map, place entities, and update/draw each grid |

|  |
| --- |
| Grid |
| //keep track of each entity within the grid  TemplateVector<Entity\*> m\_mapOfEntities |
| //various functions to clear non-player entities, add/remove entities from the grid, and update/draw |

|  |
| --- |
| Tile (struct) |
| //visual tile to draw  SDL\_Sprite \* currentTexture  //flags to keep track of whether it can be collided with, animated, or is a spawn point  //location in world space & camera pov  SPoint pos, \*cam |
| //use pos & cam to calculate its screen location  SPoint getLocationScreen() |

All member variables of a class will use m\_(variable name). Each header file should always start with a comment, explaining the reason for the file, what it does, who made it, and who edited it last. After the comments, start with #pragma once, then begin all of the #includes.

If a function is reasonably short ( < 3 lines), put it in the .h file. Otherwise, make a .cpp and put it there.

# Story of Our Problems

December 12th, 2010. The start of a new day. A group of four came together and said to each other, “Let’s make a game.” So the four went onto facebook and started a thread, detailing everything they want in the game. Their lives shown to everyone who had access. Problems, and victories, and even things not relating to the topic but just personal areas. And it’s here that this story starts.

March 4th, 2011. The group of four had come together twice every week for a while now in order to design their game. The code hadn’t started, and hopes were high. And then the news came; they had to switch engines. They had intended to use the Irrlicht engine under the direction of their lead programmer. Their modeler and artist wasn’t able to use the engine correctly, and so it was quickly scrapped in favor of using SDL.

The code started. Hopes were high, and it was quickly apparent that this was a team unlike any other. But the lead programmer, once slighted, was not able to easily forgive them. And so he didn’t do much, if any work. Distracted, he said, by various other areas of his personal life. But the others forgave him just as easily as any other.

Sickness, computer problems, all became known to the other three. An original part of the design was networking, and with none of them having previous experience, it was slow going. That ate up one of their time all the time. The other two had high hopes however. Once the basics were set up it was obvious what was next. Content.

Then more bad news came. The lead programmer, distracted as he was, left the team permanently. All was sad to see him go, but knew it was a necessary thing. He even left on his own terms, and offered his help when it was requested of him. The other three trudged on. Bugs, crashes, even files randomly changing rapidly became issues.

Then the light shone! Another member, to replace their lost one! The team of three was once again four! Hopes were again high as they went on, fixing bug after bug, adding feature after feature. The three had decided against adding in networking, seeing as none of them had previous experience. But now they did!

So one of them took on the effort once more. Not full-time, but enough to get it working in a few weeks. More content was added, more bugs were fixed and crashes erased or came up. But a few weeks was not enough for the one to finish it.

So in the time limit the four had they decided against showing it, knowing it was incomplete as it was. The deadline was soon on their heels, and not knowing what to do about it they fixed more bugs. Then it came up from behind them, and took them by storm. And it was only through their previous efforts that they were as prepared as they were. And still, the four trudged on, daring anyone to try to slow them down.

This is our story. This is the story of Team BMP.