## VGP130 / VGP 220

## Course Project

## GAUNTLET



## Thiago Alves Braga

## Stephan Botha

# Project Proposal

The game which we plan to make is based on the classic 1985 Atari arcade game “Gauntlet”. This is a top-down, hack-and-slash dungeon crawler in which the player ventures into a dungeon filled with enemies. The goal is to gather as much treasure as possible before dying. We plan to use openGL to implement 2D textures and animation.

For the C++ II class we hope to use the following concepts: Smart pointers and Operator Overloading. From the Data Patterns and Algorithm class we plan to use maps and a binary search tree. Concepts from C++ 11 which we will implement in our game include the auto specifier and ranged based for loops.

# Description of game

Gauntlet is a top-down dungeon crawler in which the player controls a warrior who has to fight his way through hordes of enemies in order to collect treasure for points. The enemies spawn from shrines on the map and will try to move towards the player.

When they come in contact with the player, the player will lose health and the enemy will be destroyed. The player can kill enemies by throwing them with an axe before they reach him. In addition shrines can also be destroyed with the axe to prevent more enemies from spawning from that particular point.

The game is made up of dungeon levels which progressively become more difficult. The exit of one level connects to the entrance of the next level.

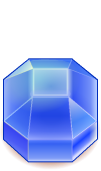
# User Manual

Keyboard commands:

* Arrow keys: Movement
* Space bar: Throw axe
* Escape: Exit Game

Health

C:\Users\Fanus\AppData\Local\Microsoft\Windows\INetCache\Content.Word\walking sw0007.pngThe player starts the game with 100 health points. Whenever an enemy touches the player he will lose 10 points of health. Collecting a heart from the map will add 100 health points to the player’s reserves. When the player’s health reaches 0 he will die and the game is over.

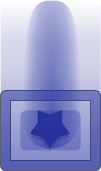


Score

The player scores 100 points for each treasure collected.

Keys

Keys found around the map can be used to unlock doors and access new areas. Beware that enemies tend to gather in large numbers behind doors!

Entrances and Exits

The player starts each level at the green map entrance. You advance to the next level by walking onto the blue exit marker. Be sure to collect all the treasure on the level before moving on to the next level, as you cannot go back to previous levels.

# Programming Concepts

C++ II: Operator overload

The == operator is overloaded in the tileClass. The function contains an algorithm to determine if the coordinates of two tiles overlap. Many of the objects in the game are derived from the tile class, for instance the player object, the enemies, the projectiles (axe) objects and the walls. During each frame we have to check for collision between all of these. By overloading the == operator, these collision checks are simplified and the code is more readable.

C++ II: Vectors

Our code makes ample use of vectors to organize the numerous game objects which are constantly being created and destroyed, including the enemies, spawn points and projectiles (gameClass.h and mapClass.h). New objects are always added to the end of the particular vector. Because objects are frequently erased from the middle of a vector, a list could have been a reasonable alternative container. However, because the objects need to be accessed randomly at times, for instance when only working with the objects currently on the screen, we elected to use vectors rather than lists.

Algorithms and Data Patterns: Recursion

A recursive algorithm is employed in the mapClass (member function findFirstDoorPiece). Doors in the game consist of multiple pieces, each of which is an object in itself. When a player with a key collides with any of these pieces, the whole door needs to be unlocked and opened. When such a collision happens the findFirstDoorPiece function recursively searches for the first piece of the door, and once found the pieces of the door are then systematically erased from the doors vector.

Algorithms and Data Patterns: Maps

A non-sequential container in the form of a map was employed to store textures objects (resourceManagerClass.h). At the beginning of the game, various textures are loaded from .png files and attached to texture objects. These objects are stored as elements in a map, and are identified by unique string keys, for instance “wall”, “treasure” and “door” ( gameclass::initializeGame() ).

C++ 11 Features

Ranged based for loops and the auto specifier were used in our code ( eg gameClass::render() ).

# Acknowledgements

* Animated sprites: [Reiner’s Tilesets](http://www.reinerstilesets.de/2d-grafiken/2d-animated/)
* Textures: [Lost Garden](http://www.lostgarden.com/2007/05/dancs-miraculously-flexible-game.html)
* openGL: [Learn openGL](http://learnopengl.com/#!Introduction)