KUHAT

C++ Programming project - Group 1

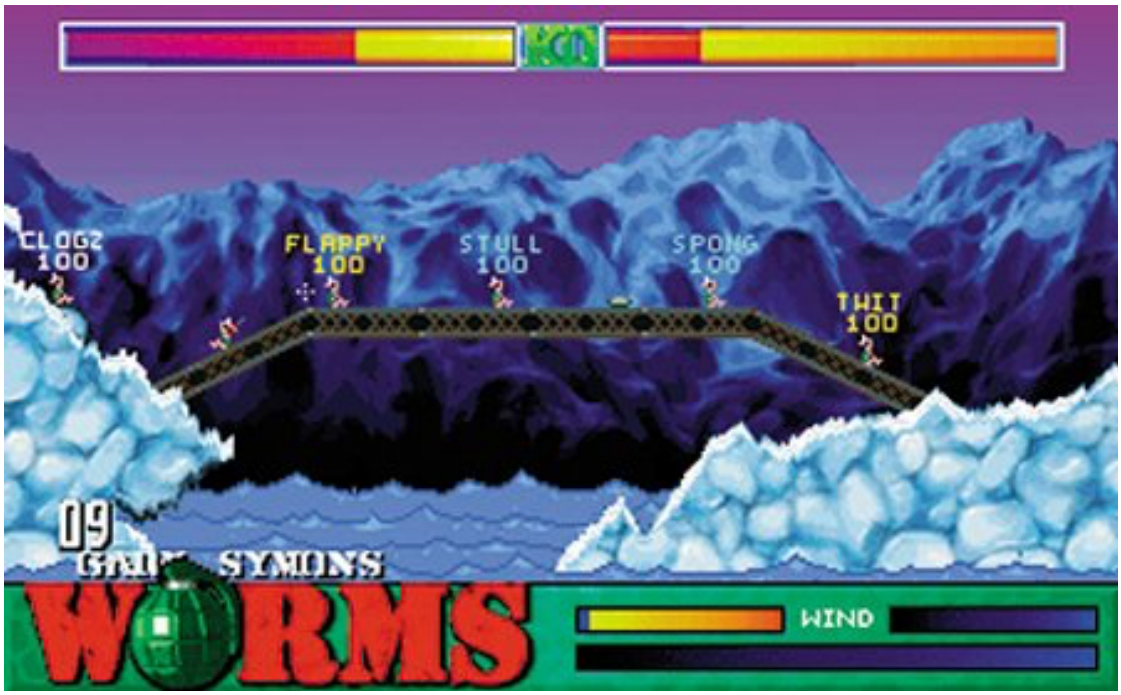
Artillery

Assigned assistant: Joel Huttunen

Team: Santeri Salmijärvi
Joel Pulkkinen
Antti-Jussi Inkinen
Juuso Mikkonen

Scope of the project

The project is based on the classic artillery game -variant Worms; A multiplayer game where players control teams of soldiers and try to shoot each other with different weapons. The characters are able to move and jump around. The randomly generated terrain is destructible but doesn't collapse on itself, creating a Worms-style Swiss cheese end result.



We have extra features in addition to the “artillery”-task in mind, which will be added to the game as time allows.

Core features:
-Characters move, jump and shoot
-Multiplayer
    -1 character per player
-Destructible terrain
-Three weapons
-Gravity and wind conditions that affect characters and projectiles
-Randomly generated maps

Possible extra features
-Team of characters per player
-Audio
-More weapons
-Health and weapon drops
-Mutators
    -Low gravity
    -High wind
    -Low health

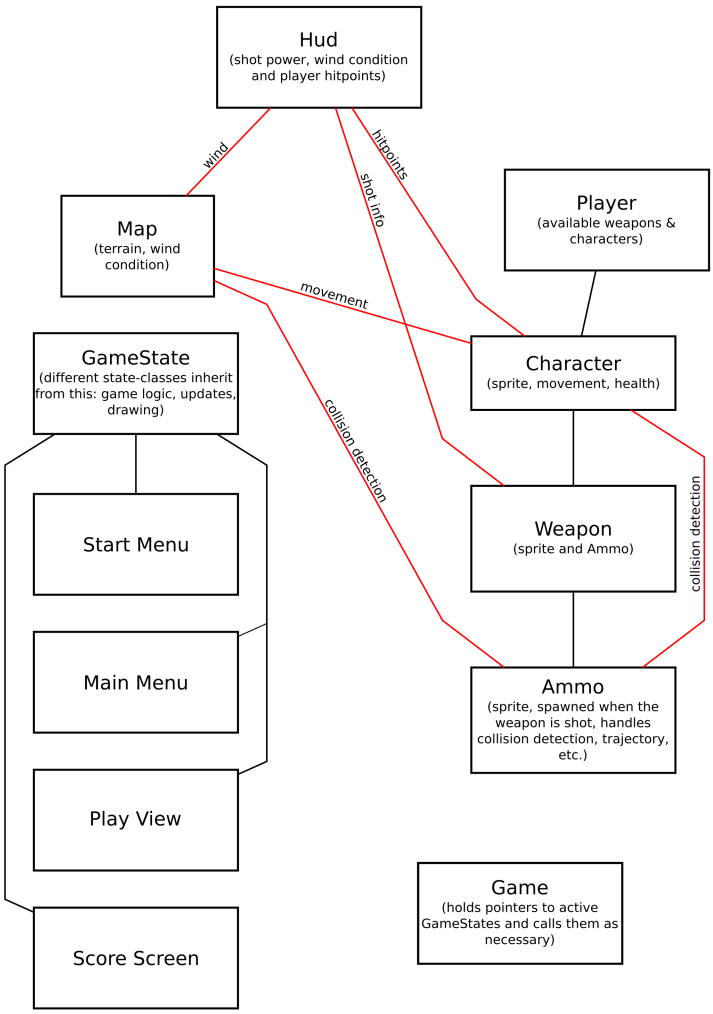
Design

SFML is used as the main library for graphics, input and audio. We aim to write portable code, so that the game should work on any platform SFML supports. Ubuntu is used as the main development platform.

Testing will be done with actual playtesting and unit tests are used when appropriate. Branches are created for all new features and only finished code will be merged to master. Master branch should always contain a working prototype.

Basic overview of the code

-Game (holds pointers to active GameStates and calls them as necessary)
-GameState (different state-classes inherit from this: game logic, updates, drawing)
    -Start menu
    -Main menu
    -Play view
    -Score screen
-Player (available weapons & characters)
    -Character (sprite, movement, health)
    -Weapon (sprite and Ammo)
        -Ammo (sprite, spawned when the weapon is shot, handles collision detection, trajectory, etc.)
-Map (terrain, wind condition)
-Hud (shot power, wind condition and all combined player hit points)



Main function initializes the game and loads the first game state. A loop then calls the update functions in Game while there is an active state. If no state is active anymore, the game has been quit and the main function will handle possible cleanup and then exit.

Code style guidelines

```
#define PI 3.14 // constants are written in uppercase
/*
 *  Classes have a description written inside a multi-line comment
 */
class Example {            // Type names start with a capital letter
public:
    /*
     *  Function/method descriptions include:
     *  @params: foo
     *  @return: none
     */
    void doStuff (Type foo) {
        return;
    }
     int fooBar = 0;          // Variables and functions use camelCase
naming
    // Indentation uses four (4) spaces, never tabs
};
```

Schedule

We plan to have a prototype game with two characters on a flat terrain ready by the mid meeting. Basic movement controls and the bazooka should work. We should have basic sprites implemented for all these components.

After the prototype is done, we will concentrate on terrain destruction and random terrain generation. As stated previously, some additional features have been planned in case we actually have time for them.