Projet C++

Bomberman

Koalab koala@epitech.eu

Abstract: **Bomberman** *is one of the most famous video game. With more that 70 franchises, since the first version on* **MSX**, **ZX Spectrum** *and* **Sharp MZ-700** *in 1983 until the lasts versions on* **PlayStation Network**, **WiiWare** *and* **Xbox Live Arcade** *commercialized in 2010 ; more than 10 millions units have been sold.*

This project results from the partnership between the **Koalab** *and the* **Game Dev Lab**. *The two teams,* **Pandas** *and the* **Koalas** *worked long hours to prepare this project. I want to use this opportunity to warmly thanks again the* **GDL**, *for their help, their total investment as well as for the quality of their work, on top of being tremendous people.*

Contents

Chapter I

Foreword

We decided to use as a reference, the video game `Neo Bomberman`, commercialized on `Neo Geo` and on `MVS` systems in 1997. It is not the best version of the series, but it holds the advantage of being specially built for two players, one against the other. Of course, it is still possible to play against the system.

I think you get it by now: we want you to code a multi player mode allowing to play against IA in your `Bomberman`. One of the main difference with the `Neo Geo` version is that graphics will be using 3D for maps and characters. However the game play stays in 2D. Feel free to test a 3D game play if you feel to. You can get inspiration from existing games.

- An overview of existing Bomberman : http://www.uvlist.net/groups/info/bomberman

We are providing you a library created by the `GameDevLab` that contains the necessary base to load and to display 3D objects. You also have access to a limited pack of 3D resources.

For most of you, this is your first video game coding experience. Perhaps you are asking yourselves how to handle all these capabilities? what to implement first? etc. For this mater, the `GameDevLab` joins the`Koalas` for this project.

We won't lie to you. Coding a video game is generally not fun. (at least not all the time.) But it actually can be..., and in this case it's pretty enjoyable to play a game you coded yourself and actually experience others enjoying your game! This result merits efforts!

So, HAVE FUN!

Chapter II

Resources

Due to the nature of the project, the GDL allocates you contents to help you with the coding.

II.1 Library

The GDL wrote a graphic library based on OpenGL for your comfort. It is **MANDATORY** to use this library for this project, with no exception!

This library is available in the repository public of the account gamelab on l'AFS :

```
1    /u/all/gamelab/public/libgdl_gl-X.Y.tgz
```

Copy the library on your local system, however you must update it regularly!

If you want to use others complementing libraries, you MUST get the authorization from the GDL team and justify your choice. It is explained later how to communicate with the the GDL.

No request will be considered until you have implemented the basics elements of the game. We will use the SVN repository to verify it.

II.2 Documentation

A good library without its documentation is a bad library. Consequently, you'll find the library's documentation in the directory textttdoc of the archive:

```
1    libgdl_gl-X.Y/doc/
```

II.3 Assets

Even if this project includes a very important graphical component, being a designer is only for an elite group of people. The GDL provides you 3D animated models for the bombs and for Bomberman. You are on your OWN if you need more models.

Graphical assets are provided to you by the GDL in the repository assets of the archive:

```
1    libgdl_gl-X.Y/assets/
```

A big thank you to Xavier "Viking" Baures for his amazing work on the models!

II.4 Tutorial OpenGL

OpenGL programing can be particularly cryptic for a beginner. In order to give you as many chances as possible, the GDL created a tutorial to help you jump start the project!

This tutorial is available at the same place that the project description on the Epitech intranet. It is essential to read all of it!

Chapter III

Mandatory part

III.1 Before anything else !

This project results from a partnership between the `Koalas` and the `Pandas` of the `GDL`.

You'll use the following forum for all of your questions: https://intra.epitech.eu/forum/#!/Modules-Epitech/Semestre-4/B4-C/Bomberman

III.2 Generalities

You MUST code a Bomberman-like, functional, and mainly inspired by the "Neo Bomberman" version:

- A complete video game with at least:
  - Intro
  - Menu, modes, options, pause, ...
  - Save and continue a game
  - Victory and defeat
  - Persistent score and ranking
- 2 players on the same keyboard

As you are using PC and considering the processors as well as the graphic power available, these are some mandatory challenges:

- The game play is 2D as for the original game, but the 3D graphics as well as animations and decent FX are mandatories.
- The number of player is unlimited. 1 OR 2 players can be handled by humans. Others are handled by the system.
- It is possible to choose EITHER a map that you build yourself OR a map generated randomly. THERE IS NO maximum size for a map. That means it is possible to play on a map of 100 cases per 100 cases, with 10 players with no slow down of speed.
- A reduction of performances is tolerated only for huge maps with thousands of players.

Figure III.1: Neo Bomberman in game

III.3 The Game

- The client MUST display a top-down view of the cases with walls and boxes... It is the map.
- Players must be moved thanks to arrows keys or equivalents for the second player (on a single keyboard). Players can also drop one/several bomb(s) and collides with obstacles (walls, boxes, Bombs.)
- Bombs MUST explodes in the four directions after a certain period of time, with a certain range and must destroy the first destructible element met.
- The speed of move and the speed of animation MUST be independent from the hardware.
- Every players of a map must have a its own skin (color-wise) You need to think about that!
- You MUST provide a solution for the case of the two human players being too far from each other to appear on the same screen.
- At the beginning of a game, each player can drop only one bomb at a time. This bomb explosion covers two cases in the 4 directions.
- Bonuses are MANDATORY within which at least the 3 following:
    - Being able to drop one more bomb.

- Increase the bomb range of one case.
- Increase the player movement of 1 unit.

- Each IA player must be scripted. Proposing more than one script can make the game more fun.
- Initial position of players MUST be random. Players MUST be as far as possible from each others.
- Scores MUST be recorded and displayed on the score board. This score board must be persistent from one execution of the game to another!
- Games MUST be saved and restored with no limits. Date as well as a small screen shot will be appreciated to identify the game easily.
- Your `Bomberman` MUST be multi-threaded thanks to the library `pthread`.
- Your `Bomberman` SHOULD have music as well as sounds effects.

The IA scripts are up to you. You can use programing languages such as `LUA` if you wish.

You are also free to use whatever you want to serialize/unserialize data that you need to save/load games or scores.

III.4 Advices

Think simplicity and with efficiency. It's a basic advice, but keep it in mind. If things gets out of control, get back, keep it easy, think again and ask questions if needed. The project is not complex, but if you want to succeed, you have to understand how the different parts are put together!

`The Epitech intranet allows you to work with groups of 4 to 6 people. We advice that a group of 6 seems a good strategy considering the work load required.`

Chapter IV

Instructions

You are more or less free to implement your program how you want it. However there are certain rules:

- It is `MANDATORY` to create an account on the forum `GDL`.
- If you figure out a technical issue with this forum, thank you to inform quickly the project assistants.
- It is `MANDATORY` to use the library provided by the `GDL`
- The only functions of the `libc` that are authorized are those one that encapsulate system calls, and that don't have a C++ equivalent.
- A library which is not explicitly authorized is definitively forbidden, within the limitations of the project.
- Each answer to a problem MUST be an object approach.
- Each value passed by copy instead of reference or by pointer must be justified. Otherwise, you'll loose points.
- Each value non `const` passed as parameter must be justified. Otherwise, you'll loose points.
- Each member function or method that does not modify the current instance and which is not `const` must be justified. Otherwise, you'll loose points.
- There are no `C++` norm. However if a code is reckoned to be unreadable or dirty, this code will be penalized. Be serious please!
- It is FORBIDDEN to have connections superiors to `(if ... else if ... else ...)`. You must factorize!
- Keep an eye on this project instructions. It can change with time!

- We are very concern about the quality of the materials. Please, if you find out any spelling mistake, grammatical errors etc. please contact us at koala@epitech.eu so that we can do a proper correction within the day.

Chapter V

Turn in instructions

You MUST turn in your project on the system provided by `Epitech`.
The repository name will be `cpp_bomberman`.

Repository will be closed at the exact hour of the end of the project, `Epitech` intranet being the reference.It is useless to complain because at your own watch you were still on time. The only relevant time is the one from Epitech which has an automatic system.

Corollary to the Murphy's law: "If you turn in your work within the last hour, something will inevitably go wrong."
Only the code from your repository will be graded during the oral defense.

Good luck!