Projet C++

Indie Studio

Koalab koala@epitech.eu

Abstract: ***Indie Studio*** *is the last C++ project of your second year in Epitech. It consists in programming a* ***3D*** *video game with* ***real*** *tools.*

You will have to choose a game to make. All your game will have to feature specific functionnalities and to offer a real gaming experience.

Contents

Chapter I

Hyper Sprint

`Sprint` is a great race arcade game, inspired by one of the first video game ever made and the first race game of the video game history: `Gran Trak 10`. `Gran Trak 10` is an arcade game created by Steve Mayer and Larry Emmons, released in 1974 by Atari. The "Sprint" serie is directly inspired by this game. They are top down race game where the whole circuit is visible on screen.

After Sprint, Super Sprint, Championship Sprint, it is time for `Hyper Sprint`. Your `Hyper Sprint` will also be a top down race game, which can of course be played multiplayer from one to at least four players (more if you manage to link more controlers to your computer). The main difference with older `Sprint` will be the addition of some `Mario Kart` features: you must add powerups on the circuit, you must hold items, you must be able to shoot, put obstacles like bananas, have turbos, etc.

- An overview of Grand Trak 10:
  https://www.youtube.com/watch?v=R014taK2pZM
- Championship Sprint:
  https://www.youtube.com/watch?v=TVzYGug3Cdk
- BadLands (Another Gran Trak 10 clone):
  https://www.youtube.com/watch?v=_5_7bFp5UMk

Chapter II

Lemmings

`Lemmings` is a famous puzzle video game. Created by Dave Jones (Blood Money, Grand Theft Auto), if was ported to almost every platform from Commodore 64 to PlayStation 3.

We will take the original `Lemmings` as reference, no "3D Lemmings".

Your game will features 3D graphics but a 2D game play with a fixed or scrolling camera.

Lemmings is a one player game.

- An overview of a clone of the original Lemmings :
  http://www.elizium.nu/scripts/lemmings/
- Listen to this music (Well... Actually it is from Menace) :
  https://www.youtube.com/watch?v=mLZYJCBAoNM

Chapter III

Worms

`Worms` is a artillery game created in 1995 by Andy Davidson and Team17, inspired by the classic video game "Artillery", by Mike Forman, released in 1976.

Our reference is Worms 2/Worms Armaggedon, so we want you to make a 3D game with a 2D gameplay.

Your game must be multi-player, but we must also be able to pay alone against the computer. Considering that Worms is a turn-based game, your game will not limit how many worms or team can be ingame.

You have to include randomly generated maps in your project.

- An overview of the gameplay :
  https://www.youtube.com/watch?v=xKrwbu8W8Pw

Chapter IV

Bomberman

`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.

We decided to use as a reference, the video game `Neo Bomberman`, commercialized on `Neo Geo` and on `MVS` systems in 1997.

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.

You have to include randomly generated maps in your project.

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

Chapter V

Gauntlet

`Gauntlet` is a great arcade game conceived by Ed Logg and edited by Atari in 1985. Gauntlet features a multi-player game playable from one to four players. It was ported to Amstrad CPC and a bunch of other machine. Today, you can play "Gauntlet: Slayer Edition" on Steam.

In `Gauntlet`, you can explore dungeons, get keys and treasures, throw spells and shred monsters. Make a 3D version, still with a top-down view of it, still multi-player, for one to four players.

- An overview of Gauntlet:
  https://www.youtube.com/watch?v=Kp9Mo9QuV2w

Chapter VI

Post-Scriptum

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. Do not hesitate to come and ask questions to members of the Koalab.

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 VII

Ressources

VII.1 Library

You can choose between `Ogre 3D` and `Irrlicht` libraries. For any other library, please ask before doing anything. You can find us on the school social network.

http://www.ogre3d.org/

http://irrlicht.sourceforge.net/

Some of these libraries cannot be installed on the standard dumps of the school. You should then consider that you will have to install manually these libraries, without privileged rights. You will acquire more than culture at this point.

VII.2 Assets

Your game must use assets. We do not want fighting cubes!

http://www.ogre3d.org/tikiwiki/Free+Resources

There are plenty of free assets on the Internet. You can of course create your own, but please keep in mind that we will judge your program.

Chapter VIII

Deployement

VIII.1 Linux

Your program must be able to be installed on a brand new dump. Your Makefile must, if launched with "install" as order, install libraries and game on the current computer.

VIII.2 Windows

Your program must include a .exe file that install your game on Windows. There is several software like install maker that will allow you to do it easily. The hardest part will be to compile your program on Windows.

Chapter IX

Mandatory part

IX.1 Generalities

You MUST code one of the previously described games. It MUST be functional and features:

- A complete video game with at least:
  - Intro
  - Menu, modes, options (Like managing controllers!), pause, ...
  - Save and continue a game
  - Victory and defeat
  - Persistent score and ranking
- At least 2 players on the same keyboard for multiplayer-games.
- You must keep the original gameplay of the game you choose, at least in outline... but turn it into 3D, with nice assets and fx.
- The original view of the game must be kept, at least in outline.
- A reduction of performances is tolerated only for huge maps with a lot of players or NPCs.
- On multiplayer Game, your program must at least handle two human players. Non played characters must have the option to be AI controled. Think about allowing a game only with AI.
- Different players MUST display different colors.

- If you choose to not include all the game inside a single screen, you must think about a way to display correctly areas around players and players themselves.
- At least 3 powerups must be available for Bomberman, Gauntlet and Hyper Sprint. At least 5 tools or weapons must be available in Lemmings and Worms.
- AI elements must be scripted. You may propose more than one behaviour.
- You must record scores and display them 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 game MUST be multi-threaded thanks to the library `pthread` or `std::thread`.
- Your game MUST have music and sounds effects.
- Your game must display your splash screen. A splash screen is a cool thing. Think about it.
    - Atari: https://www.youtube.com/watch?v=3THphLEveBA
    - Activision: https://www.youtube.com/watch?v=ZBeuWn1NFZ4
    - Bullfrog: https://www.youtube.com/watch?v=gUoofVDXKQY
    - Sega: https://www.youtube.com/watch?v=B3mMY-_n3_E

Figure IX.1: `Neo Bomberman in game`

IA scripts, if required by your game, 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.

IX.2 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 X

Instructions

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

- 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 XI

Turn in instructions

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

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!