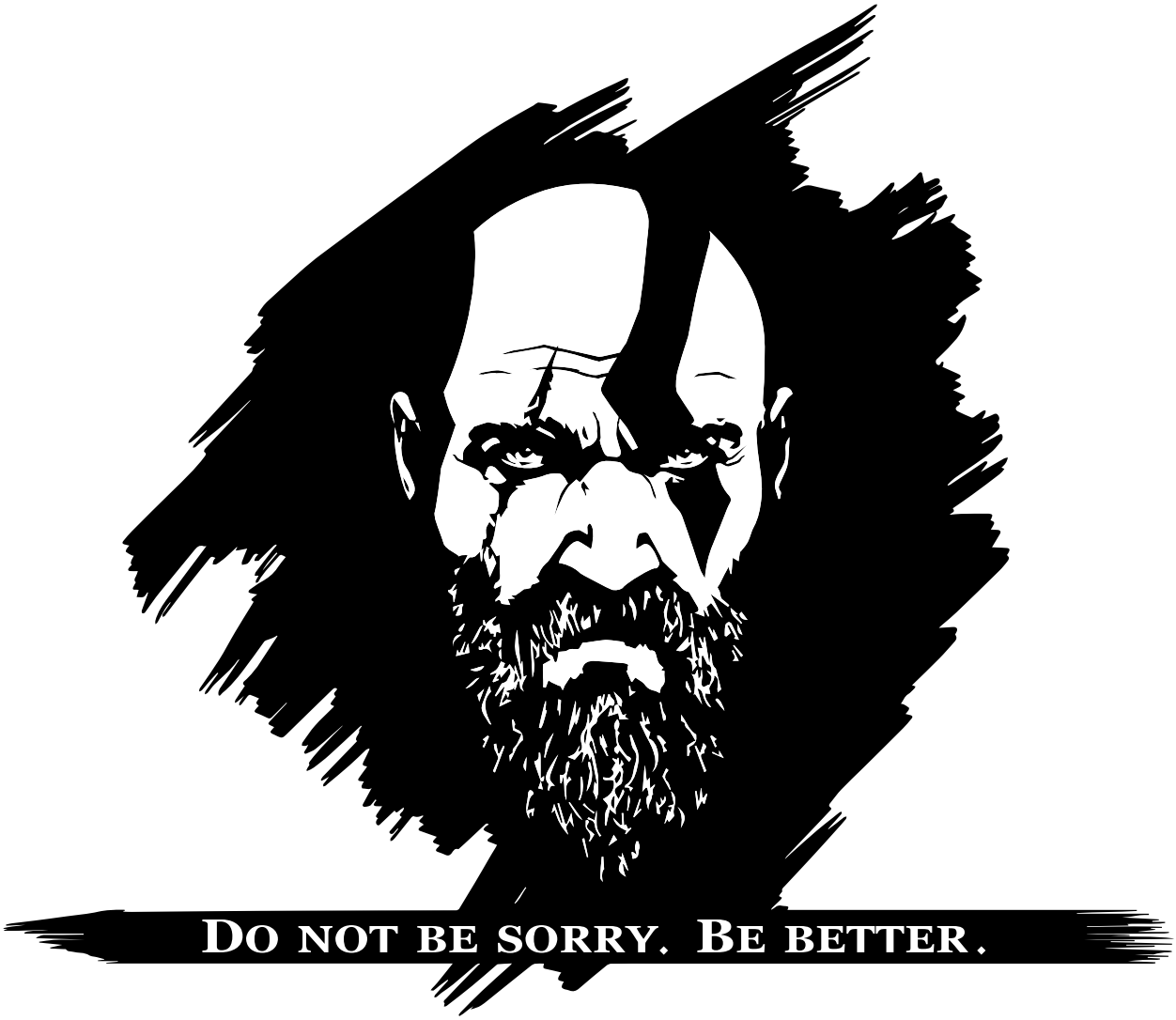
RUSH C — Subject

version **#deploy-rush-c-2021-v0.3**



ASSISTANTS C/UNIX 2019 <assistants@tickets.assistants.epita.fr>

Copyright

Contents

*. https://intra.assistants.epita.fr

Project data

Instructors:

- Sébastien Rouet <rouet_s@assistants.epita.fr>
- Edouard Veyrier <veyrie_e@assistants.epita.fr>

Dedicated newsgroup: assistants.projets with [VGR]

Members per team: 3

1 rush-c

Files to submit:
— `./src/*`

Forbidden functions: none

Allowed libraries: SDL2

2 Introduction

As you already know, you will create a video game during this weekend. The rules stay the same for each rush:
— 38 hours
— Hard work
— Team project

2.1 Evaluation

— Your project will only be tested during your defense.
— Pay attention to the C coding style.
— Exceptionnaly, body functions can have up to 50 lines.
— You must build your project at the root of your repository with a Makefile or a build system (Cmake, Autotools).

Feel free to do whatever you want as features and most of all... have some fun! The main objective for you (and for your grade) is to build a cool game while implementing as many features as you can. Keep in mind that simple games can be really fun whereas complex games can be annoying; it's your choice.

We will evaluate the gameplay but also technical elements:
— Graphical and physical engines
— Input handling
— Ergonomics
— Game features
— ...

We will give you a vague description of what we expect as mandatory. This will not be sufficient to get a good grade, and don't expect us to tell you which grade you will get if you only do the mandatory part...

Keep in mind that having a working game must be your **priority**: the graphical facet of your game must come after.

2.2 Theme

We spoke about grades, features and fun, but we still didn't say what your *chef-d'oeuvre* will be about...

Creating a **platform** game!

Platformer is a kind of game where, as its name stands for, the player interacts with platforms to reach the end.

Don’t hesitate to have a look to old games or indie games to get ideas of what you might do!

Please take note that the subject is evasive to enhance your creativity. You are free to make the game you always dreamed for as long it matches the subject requirements!

This mechanics can be found in the following games:
- Super Mario World
- Donkey Kong
- Braid
- Rayman Legends
- FEZ
- Super Meat Boy
- Mega Man 2
- Spyro
- Sonic 3 & Knuckles
- Wario Land 3
- VVVVVV
- Ghouls ‘n Ghosts
- Limbo
- I Wanna Be the Guy
- Asterix (1993 video game)
- Cuphead

3 Mandatory

Your display *must be* done using **SDL2** or **OpenGL**.

However, if you are not already familiar with OpenGL, we **strongly** advise you to use the SDL2 library.

The main directive is to have a playable game that can be restarted when loosing. To get the hang of the platformer mechanics, the game shouldn’t be too easy and you should die a few times .

Below are guidelines that will help you out by breaking down different entities of the game we expect you to create. The implementation is up to you, but you must at least include these elements.

3.1 Player

You must have a character who is able to move and complete a set of actions to win the game. At the very least, the main character should be able to move properly.

3.2 Rooms

The character will evolve through maps/rooms. However, you can have multiple maps/rooms that are connected through their exits but this is listed as an additionnal feature. You can see below two constraints that we want you to follow.
- You are not allowed to hardcode your maps and enemies. You must come with a way to modify easily the content of your map and enemies disposition, and so on (which means you cannot put ennemies/player position directly in you code).
- You must have at least two different maps/rooms.

3.3 Enemies

Your game must have at least one kind of enemy and another way to harm the player.

Beware that we are evaluating the differents mechanics you are able to make. For instance, falling on spikes and falling in water count as only one danger, as the game mechanic is exactly the same for both.

3.3.1 Opponent

Because a game without challenges is pointless and boring, you must add enemies in your map/room. They must be able to harm the player, and they must be endowed with the behavior of an... artificial intelligence, even a dummy one.

3.3.2 Danger

We ask you to have at least another danger in your game. It could be another kind of ennemy, traps, spikes, lava or anything else. As long it can harm the player, you're only limited by your own imagination!

3.4 Game over

The concept of victory and defeat must be clear. Furthermore, to stick to the platform genre, your game can have a way to start over without exiting the game, particularly after dying, with some checkpoints for example. Keep in mind that we do not want to see your game end abruptly. You must show explicitly that the player has lost or won. It could be an ending screen, a sound, or something else.

During the defense, we must see that your game seems possible to complete. Surprise us!

4 Additional features

The features are quite important in this project, they will allow you to reach a good grade and, more importantly, to make your game unique and fun.

If some features you implemented don't match any points below, and you think it is worth consideration, you can implement it and add it to your list of features.

However, if it requires additionnal arrangement for the defense, you *must* inform us by using the dedicated newsgroup *before* the deadline, with an accurate description. Think carefully before sending your news.

> *[VGR][NAF] Feature title*

Here is a non-exhaustive list of features you can implement for your game, sorted by difficulty. Note that you do need to list all of your features in you `README`, wether they are in the subject or not.

These features can be implemented in most video-games.

Some of them are very basic things we expect to find in a video game, like sound, or animation and we advise you to implement many of them as it will increase the overall quality of your game. Obviously enough, harder features will give you more points.

4.1 Individual features

4.1.1 Easy

- **Sound/Music**
  You must have at least two types of music that can change, for example when moving from a map to another.
- **Title screen**
  "This tower is so huge, we even had a loading screen back there" - Pit, Kid Icarus: Uprising.
  Lauching your game must display a title screen, with at the very least a launch and quit button.
- **Score**
  "The numbers Mason, what do they mean?" - Call of Duty: Black Ops.
  You must display during the game the score at every moment of the game. The way it increases depends on your game.
- **Timer**
  A timer is a countdown that add some challenge! If the timer is up, an event must happen.
- **More than one way in the game design to beat the game**
  "The ending isn't any more important than any of the moments leading to it." - Dr Rosalene, To The Moon.
  At one time in the game the player will face one or multiple choices. Based on the result of these choices the way to win in the game is different. These feature cannot be validated in randomly generated worlds
- **Invincibility frames**
  After taking damage, there is a small amount of time during which the hero cannot be damaged. There must be a way to see that the character is invincible during this time.
- **Hero evolution**
  "The only thing that can defeat power, is more power." - Albert Wesker, Umbrella Chronicles.
  The hero can evolve in a given way (more stats, faster, life up, ...)
- **Delta Timing**
  Implement the delta timing as introduced in the presentation. ie: Synchronize the time in the game world independently of the computer you run your game on.
- **Animated environment**
  At least two elements of the game environment must be animated.
- **Permanent leaderboard**
  Scores attained during the game are sent to a server. This server must be able to display a leaderboard.
- **Difficulty levels**
  The user can change the difficulty of the game. This difficulty change must influence at least one charateristic of the enemies or their number. You must be able to finish your game in every difficulty level.
- **Map powerups**
  Some items can be found on the map. These items, once taken by a character, must have an effect.

4.1.2 Medium

- **Human dubbing**
  "..." - Link, The Legend of Zelda (any of them).
  Add some human voices to your game! However, it must be consistant with your game, which means a voice dubbed shoutting something at a random moment doesn't validate this feature.
- **Scrolling**
  In some kind of games, a scrolling is unavoidable. Your scrolling must work properly (even at the edge of the map). The quality of the scrolling will be evaluated (starting to scroll when the character is at the center of the screen isn't a good choice in almost every game).
- **Cheat codes**
  When you press a button, a text box allow you to enter input text. If proper input is given, something that helps you to finish the game or funny happens!
- **More enemy kinds**
  In addition of the mandatory part, this feature implies that you have a new ennemy kind with behaviour that differs from the previous one (just swapping sprites doesn't work).
- **NPC dialogs**
  "Stay a while and listen." - Diablo II.
  Non player characters with whom you can talk with.
- **Gravity**
  Isaac Newton would be proud of you.
  A feature that must exist in a plateform game. Your player must fall if there is no block under him.
- **Jumps**
  "Do a barrel roll!" - Star Fox 64.
  You must give to your character the ability to jump, which is in general paired up with the gravity feature.
- **Loot**
  The enemies must have a probability of dropping items that give powerup to your hero (often enough to be seen during the defense).
- **Random dynamical loading**
  Parts of the levels are dynamically loaded. These parts must either be randomly generated or randomly chosen from a set of map.

4.1.3 Hard

- **OpenGL shaders**
  Try to get a better rendering using openGL shaders. If you have done this feature, you *must* post a news about what you have done using Opengl.
- **Boss**
  Higher level of difficulty, at least two attack patterns, breaks the current flow of the game and unique sprites. For example: you are in a closed room and can't advance until the boss is dead.
- **Resource system**
  "Pills here!" - Left 4 Dead
  Ability to buy items or powerups in game with gold, points, and so on. Items must be useful.
- **Replay**
  Able to record your run of the game and store it in a video file.
- **Advanced NPC**
  "It's dangerous to go alone! Take this!" - Old man, The Legend of Zelda.

Non player characters that helps your hero with smart enough AI to be useful.

— **Multiplayer**
“We stay as a team. None of that lone wolf stuff” - Halo: Reach.
Able to have two or more players.

4.2 Incremental features

In the incremental features category, only the biggest level done in every group of features will be taken into account.

4.2.1 Improved Enemies

— **lvl 1: Enemies can die**
“It’s super effective!” - Pokemon series.
Your character is able to fight enemies and kill them. Enemies only have a state alive or dead. We must see that your character is actually fighting.

— **lvl 2: Enemies have a health bar**
Ennemies have a quantifiable amount of health that will be visible through a health bar.

4.2.2 Animation

— **lvl 1: Animated Character**
“Sonic’s the name, speed’s my game.” - Sonic.
Your character sprite is animated and change depending on the action of your character.

— **lvl 2: Animations**
Your character and enemies’s sprites are animated and change depending on their actions. To validate this feature, the main character must have a sequence of sprites that makes his actions look smooth.

4.2.3 Various ways to die

— **lvl 1: Various ways to die**
“Requiescat in pace.” - Ezio Auditore, Assassin’s Creed II.
This feature is validated if you have more than one extra way to harm the main character in extra of the mandatory part. This feature cannot be validated with a new enemy kind (see the *more ennemy kinds* feature instead).

— **lvl 2: Various ways to die**
“Finish him!” - Mortal Kombat announcer, Mortal Kombat.
The extra way to die must be animated in a way that it is dangerous for the main character.

4.3 Group features

In the group features category, each feature in every group of features has value. If you manage to implement every feature in a group (such as Rogue like), you will get additional points.

4.3.1 Level randomization

To extend your game and surprize the player, we designed a few set of feature if you want to make a game with some random generation.

The features aspects revolve around using items and maps that are not hardcoded, the rest being up to you

- **Random map generation**
  "There was a HOLE here. It's gone now." - In-game text, Silent Hill 2.
  This feature is the core, so it can count as an easy, medium or hard feature depending on the quality of your algorithm of map generation. Please note that a randomly generated map counts as more than 1 map. Thus, a randomly generated map counts as two levels in the mandatory part.
- **Custom events**
  Some randomly generated rooms or events that contains traps, great amount of monsters, treasure, ...
- **Differents levels**
  "Thank you Mario! But our Princess is in another castle!" - Toad, Super Mario Bros.
  You have several transitions between your game before actually reaching the end. It means the random generation must be launched back again during the game.

4.3.2 Fog of war

Adding a fog of war is a simple way to make your game sexier!

- **Simple fog of war**
  "No matter how dark the night, the morning always comes." - Lulu, Final Fantasy X.
  The rooms not yet entered cannot be seen, or the character's field of view is taken in account.
- **Déjà-vu**
  I've just been in this place before!
  Places where you've never been to, places where you have already been and where you actually are must have different intensities of visions. For example places where you have never been can be black and places where you have already can be gray.
- **Torches**
  "I should have been the one to fill your dark soul with LIGHT!!!" - Devil May Cry
  Some objects in the room, torches for example, also give sight around them.

4.3.3 AI

Artificial intelligence, because a game with some challenge is funnier.

- **Medium advanced AI**
  A medium advanced AI will be granted if you have an AI that does more than one action (going left, then right is just considered as moving).
- **Advanced AI**
  "Your entire life was a mathematical error, a mathematical error I'm about to correct." - GLaDOS, Portal.
  An advanced AI will be granted if you have an AI that does Smart actions like pathfinding or specific killing mechanics depending on range.

5 Complementary information

5.1 Tips

Here are a few tips on how **not** to fail your rush:

- The first thing you should do before opening your editor is discussing the project with your team. Be sure you have a common goal.
- Then you *must* split the work. Your timing is short, so everyone has to work hard.
- One step at a time, the first one is the *mandatory* part.
- Read the documentation of the SDL2.
- We are not asking you to become designers in one weekend so as long as your graphics don't make us blind, you must not focus on these.
- Be fully prepared to explain the features and usage of your game during the defense! Your ACU will not guess how to play!
  There must be a file called `README` that must contain:
  - The steps to follow to build your game.
  - The steps to follow to play your game (controls, story, goals, etc).
  - The sources used.
  - The list of features to test.

  Since every game is unique, explain your game well.
- Remember every feature you want to show us in order to avoid missing one.
- Lack of sleep leads to stupid errors.
- Use data structures (vector, lists, ...)

5.2 Game loop

A game can be seen as a simple loop that runs as long as the game is on. The game loop is composed of 3 steps:

5.2.1 Events

In this step, the events are caught such as the user's keyboard inputs.

5.2.2 Update

In this step, the elements in the next frame of the game are computed depending on the previous game information and the potentials events caught.

5.2.3 Draw

In this step, the new frame gets drawn depending on the new game information computed with the *Update* step.

5.3 Graphics

You are here to code in C, so we won't judge you only on graphics. But keep in mind that your grade will depend on your ACU's enjoyment. You can find sprite sheets on the Internet (it's not cheating as long as you quote the sources) and you may also animate your characters! A good sprites website can be found here https://www.spriters-resource.com/

You are allowed to use any music or sprite you like for your game.

5.4 Vocabulary

5.4.1 CPU & GPU

In order to clarify the use of surfaces and textures, you must first understand what are the differences between CPUs and GPUs.

- A CPU (Central Processing Unit) is the processor of a computer, the electronic piece of hardware that will execute instructions.
- A GPU (Graphics Processing Unit) is the piece of hardware in charge of producing the output image needed by the display devices.

As GPUs are designed specifically for fast memory manipulation and alteration, they are a lot faster than general purposes CPUs for computing frames, etc. The CPUs will be able to do those same tasks, but slower.

5.4.2 Surface & Texture

Now that you know the difference between CPU and GPU, you understand that we will aim to make our frame computations in the GPU, as it is faster. To do so, we must put our image in the GPU memory.

But we cannot directly do that. We have to first load our image in the CPU memory. That's what a surface is.

- A surface is the memory representation of your image file into the RAM, your CPU memory.
- A Texture is the memory representation of your image file into the GPU dedicated memory.

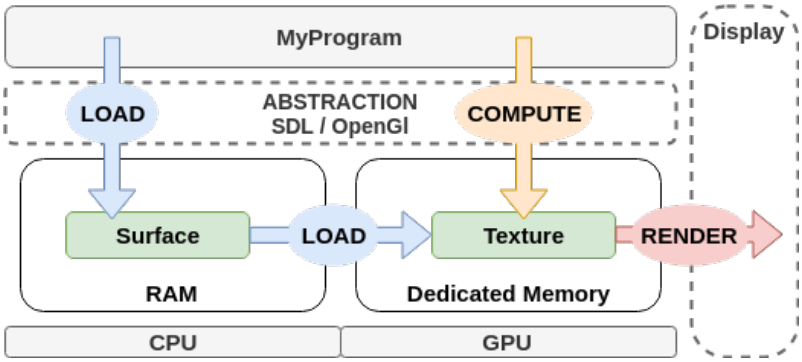
Once it is loaded in GPU memory, you can make fast computation on it.

SDL/ OpenGl will try to load the texture in GPU memory. If it can't do that (if you do not have GPU for example), it will keep it in RAM and simulate the GPU behavior. This way, you don't have to worry about this specific case while using its API.

5.4.3 Rendering

As you have seen in *Raytracer*, rendering is the act of computing an actual image output from memory objects. The renderer will simply be the program that will do this job (see *SDL_Renderer*).

5.4.4 Recap



5.5 Physics 101

In your game, you are supposed to manipulate physics to handle your positions, collisions and other interactions.

5.5.1 Reminder

All of the following formulas are expressed in function time *t*.

The acceleration is the sum of applied forces divided by the mass of an object.

$$Acc(t) = \frac{\sum F(t)}{mass}$$

We also know that:

— the acceleration is the derivative of the speed, ie the evolution of speed over time.

$$Acc(t) = Speed'(t)$$
$$Acc(t) = \frac{\Delta Speed(t)}{\Delta_t}$$

— the speed is the derivative of the position, ie the evolution of the position over time.

$$Speed(t) = Pos'(t)$$
$$Speed(t) = \frac{\Delta Pos(t)}{\Delta_t}$$

Hence, the three equations of motion:

$$Acc = \frac{\sum F}{mass}$$
$$\Delta Speed = Acc \times \Delta_t$$
$$\Delta Pos = Speed \times \Delta_t$$

5.5.2 Application

Once you understand the previous formula, it becomes very simple to compute the changes.

Most commercial games use an integrator called the *semi-implicit euler* integrator to compute as accurately as possible those variables.

All it does is simply update the acceleration, then the speed, and finally the position in this very order. The order is important to minimize the error margin as much as possible accross all iterations.

All that is left is to use the delta time as follows:

```
// Update acceleration
vel += acc * delta_time;
pos += vel * delta_time;
```

5.5.3 Links

Here is an interesting link to a presentation on how to compute a good jump.
— https://www.youtube.com/watch?v=hG9SzQxaCm8

6 Conclusion

The main objective of this C rush is for you to use everything you learned so far in C and have a result in 38 hours. Think about your design before starting to code. This weekend is also a way for you to show us that you are creative, have a lot of ideas and know what teamwork means! You must feel proud of your work at the end of the weekend, so work hard and enjoy.

Do not be sorry. Be better.