DEVELOP AT UBISOFT

Develop at Ubisoft 2023-2024

Programming: Challenge Brief

The theme of this challenge is **"Limited Resources" – it is suggested to use 1 input button in addition to direction inputs.**

The theme – Limited Resources – is a starting point, and we encourage you to explore and be creative. You can use one or multiple resources – your choice! Show us your interpretation and imagination of *limited resources* – does your resources deplete continuously or do your player's actions affect the resources?
You are allowed to map inputs to a single action (e.g., jumping, shooting, flipping gravity etc.), or multiple actions to a single button. Your inputs can also be contextual. E.g., multiple actions could include adding contextual actions based on where the player is or firing/reloading a weapon based on ammo count.

This is a technical challenge, so we will ignore artistic merit in the judging. We've intentionally kept the rendering limited because we want to see how well your simulation works, rather than how good you are at drawing. However, it would be nice if you show us some fun and creativity in your game.

This challenge must be done individually, team-based submissions will not be reviewed. All the code you submit must have been done by yourself, however, you are welcome to use the standard library.

You are **not** allowed to use external frameworks in this challenge (e.g. GLM, rendering libraries etc.). You can **only** use the standard C++ libraries. **Do not** edit the API provided (any files inside the *App* folder). If you have any questions about the rules of using the API, feel free to email us at dau@ubisoft.com

Development Environment

Your entry must be a Windows application written entirely in C++. You will need to download Microsoft Visual Studio 2019 or 2022 to successfully complete this challenge. We recommend Visual Studio Community 2022. If you're a Mac user, please use Boot Camp to install Windows 10.

Click here to download the most up-to-date version of the API

DEVELOP
AT UBISOFT

Please note the following updates to the API (since the last release):

Issue	Fix
Sprite Animation accepts frame numbers outside of acceptable range	In *CSimpleSprite::SetFrame*, update the range check to use *>=* instead of just *>*
Sprite animation time calculations incorrect when delta time is large	Use *fmod* in *CSimpleSprite::Update* to correctly compute animation time
Texture loading using c strings in map keys	Update the *SimpleSprite* class to use C++ *std::string* as the loaded texture map's key type for flexibility/robustness
When starting a new animation on a sprite, it won't start from the beginning	Add an overload to *CSimpleSprie::SetAnimation* to allow starting the new animation from the beginning. This keeps the default behaviour & sample code working while allowing a fix for the issue
When pressing the Quit input key the Shutdown function is not called.	Update the *Idle* function inside *main.cpp* to not hard exit the application using *exit(0);*, but instead just leave the GLUT loop by using *glutLeaveMainLoop();*

Guidelines

- Be sure to follow all instructions regarding file formats.
- Feel free to be creative and push your game in an interesting direction.
- **DO NOT** spend too much time on the aesthetics. Remember, this is a technical challenge!
- Use **your time effectively.**
  - One step at a time: start small to reach something big.
  - We encourage you to comment out code for things that didn't make it into your game instead of deleting it (we can still review it).
  - Leave time for testing your game & recording/publishing your video before submitting.
  - Give yourself time to complete the submission form.
- When you reach a stable state in your work, we advise you to save your project to be able to revert to it in case you get stuck. We recommend using a source control solution such as Git, or even make a local copy of your code to serve as backup.

Tips

- High level class comments are appreciated;
- In code comments to explain why, not what;
- Function signatures should communicate a lot about intended use;

- Have a clearly defined data ownership model;
- Make use of classes & structs to organize functionality and state information;
- Use appropriate C++ utility classes; E.g., Use a vector rather than C-style array where appropriate;
- Smart Pointers significantly ease memory management and code readability;
- The C++ standard library provides many useful classes to help you. Before starting to code something that seems generic, check to see if it is already provided there;
- C++ and algorithm usage can support easy to read extensible code;

Judging Criteria

Your submission will be scored out of 30. Here are the points we will consider:

Code Structure: (1-15)
- Clear and simple extensible code
- Self-documenting code with clear interfaces
- Well-managed memory and data flow

Technical challenge: (1-10)
- Usage of modern C++
- Applicable algorithms and data structures

Playability: (1-5)
- For example: AI behaviors and character control fluidity, etc.
- Fun game
- Adheres to the theme

Reminder: This is a technical challenge, so we will ignore artistic merit in the judging and focus mainly on your programming skills.

Submission Requirements & Documentation

Complete this Submission Form where you will be asked to provide a link to your submission package and YouTube video.

Deadline - **Sunday, January 14, 2024, 11:59 PM**

Your submission package must contain the following two (2) parts. Both parts are mandatory:

1. Your complete code and any documentation saved in a .zip file.
    - Documentation must be in PDF format and should be succinct, clear, and readable.

2. The link to a YouTube video of a play-through of your game with screen capture software (e.g., OBS) highlighting the gameplay and technical features. Please make sure the video is of good quality. Your video **should not** be longer than 5 minutes.

Questions?
Email us at dau@Ubisoft.com. Note: Please try to email us your questions by 4 PM EST on January 12, 2024.