Collège LaSalle
Montréal

Final Evaluation: 40%

Course Identification

Name of program – Code:	COMPUTER SCIENCE TECHNOLOGY – VIDEO GAME PROGRAMMING – 420.BX
Course title:	**GAME ENGINE III**
Course number:	420-J19-AS
Group:	07328
Teacher's name:	Émile Fréchette
Duration:	3 hours
Semester:	Fall 2024

Student identification :

Name: _____ Student number: _____

Date: Dec. 9th, 2024 Result: _____

☐ I declare that this is an original work and that I have mentioned all sources of the content that I am not the author of (online and print sources, images, graphics, films, etc.) in the required citation style.

Standard of the Evaluated Competencies

Develop native applications without a database - 00SR

- Program the application logic
- Control the quality of the application
- Participate in the deployment of the application
- Produce the documentation

Develop non-transactional Web applications – 00ST

- Analyze the application development project
- Prepare the computer development environment
- Control the quality of the application
- Produce the documentation

Develop gaming or simulation applications - 00SW

- Control the quality of the application
- Participate in the deployment of the application

Guidelines

- **Permitted equipment: calculator, computer, course notes, internet forums, etc.**
- **No breaks will be granted during the exam. No student may leave the examination room until half of the allotted time has elapsed. A student who leaves the examination room may not return to the examination room (AEIP – Article 5.12.4).**
- **The teacher will not answer questions.**
- **Students are not allowed to communicate with each other.**
- **The teacher has the duty to identify language errors in students' work. A percentage attributed to the language is up to 20% of the score (AEIP – Article 5.7).**
- **Plagiarism, attempted plagiarism or cooperation in plagiarism during a summative test result in a grade of zero (0). In the event of a repeat offence, on the same course or in another course, the student is awarded a "0" for the course concerned. (AEIP – Section 5.16).**

Points

- **1 scenario worth 60 points**

TOTAL: **60 POINTS**

Details

For the final exam, you must rework the provided Unreal Engine 5.4 project, titled “Bob and Alice” to address the issues outlined below. When you are done with your changes, submit your modified version of the project using an Azure DevOps repository with minimal content only.

This project contains native code, you will need to launch it through an IDE.

Bob and Alice live in two different dimensions connected by an inter-dimensional portal. These dimensions are represented by two Unreal Engine levels (L_Alice and L_Bob). Alice wants you to deliver a package to Bob, however your current version of the project has some issues which prevent this delivery from occurring.

The game starts in L_Alice. The player should grab the package and deliver it to Bob's house (L_Bob) in 45 seconds or less to win.

ISSUE 1

Open `L_Alice` and play it, you will see a package. The first step of the exam is to modify the existing blueprints **to make it possible to collect the package**.

The package should be collected on contact with the main player's `Capsule` by using its implementation of the `BI_InterfaceInteraction` interface.

Alternatively, if you are not able to use the existing interface, find another way to make the package disappear and update the HUD and GameState when the package is collected.

ISSUE 2

There is a locked gate preventing you from exiting the house in `L_Alice`. There are two problems with this gate. First, the buttons don't work. You must find a way to call the GameMode's function `RequestEnterGateCode` when a gate's button is pressed. This function is already implemented for you.

Then, you must finish implementing the function called `_ValidateGateCodeCorrectness` in the Game Mode. This function should compare the current key code with the code answer. Both values are within the game state. It returns true if the entered code matches the answer.

ISSUE 3

While the package can be collected, when you use the inter-dimensional portal connecting L_Alice to L_Bob, you will notice that **the state of the game is not preserved**. The values which represent whether the package has been collected or not, delivered, as well as the remaining time, will ALL reset when moving across levels. The state of the game should be

continuous (if 34 seconds of time are left when moving through the portal, 34 seconds should be left when appearing again, etc.)

Find a solution to this problem which should use the native game instance subsystem pre-created for you: `UBobAndAliceGameSubsystem`.

If you cannot use C++, you may use a blueprint GameInstance to complete this issue.

ISSUE 4

Delivering the package should be done using the 'E' key on the keyboard, you will need to implement this. A special sphere collider called `DeliverRadius` is placed on the custom exam Pawn to this effect.

The pre-made function on the exam GameMode called `RequestDeliverPackage` should be called during the input event only if:

1. The actor `BC_Chest` is overlapping the `DeliverRadius` sphere collider AND
2. The package is collected AND
3. The package is not already delivered

Additionally, the logic of this function is broken, and it should be fixed. The intent of the function is to do this sequence of execution:

1. Mark the package delivered.
2. Update the UI

You can use the function `RequestCollectPackage` as an example on how to finish implementing `RequestDeliverPackage`.

Good luck! Complete as many of these issues as you can! And do not forget to submit an Azure DevOps repository with minimal content (Git LFS is needed to properly store the assets online).

It is part of the grading expectations that each issue be committed individually to the repository with a clear message indicating what has been fixed. (Competency skills: "Proper identification of the information to be written up" and "Compliance with issue tracking and version control procedures")

EVALUATION CHART

Skill	Name	Weight
00SR.4	Proper programming of interactions between the graphical user interface and the user	8
00SR.4	Proper programming of communications between the peripheral devices and the software functions of the target platform	6
00SR.4	Precise application of secure coding techniques	4
00SR.7 + 00ST.9	Clear record of the work carried out	6
00ST.1	Proper identification of the tasks to be carried out	4
00ST.2	Proper installation of software and libraries	2
00ST.2	Appropriate configuration of the version control system	2
00ST.2	Proper importing of the source code	2
00ST.7 + 00SW.5	Precise application of test plans	2
00SR.5 + 00ST.7 + 00SW.5	Thorough reviews of code and security	6
00SR.5 + 00ST.7 + 00SW.5	Relevance of the corrective actions	8
00SR.5 + 00ST.7 + 00SW.5	Compliance with issue tracking and version control procedures	6
00ST.9	Proper identification of the information to be written up	4

/60

CORRECTION GRID FOR LANGUAGE

Clear Communication	Clear Communication, **most of the time**	Vague Communication	Unclear Communication
- 0	**- 0,5**	**- 1,5**	**- 2**
(Word Choice) Use of precise and rich vocabulary	(Word Choice) Use of precise vocabulary	(Word Choice) Use of imprecise vocabulary	(Word Choice) Use of inappropriate vocabulary
- 0	**- 0,5**	**- 1,5**	**- 2**
(Format/Type of work) Respect of norms	(Format/Type of work) Respect of **most of the** norms	(Format/Type of work) Non-respect of the norms	(Format/Type of work) Inappropriate in relation to the required norms
- 0	**- 0,5**	**- 1,5**	**- 2**
(Linguistic Code) (≤2 mistakes / page)	(Linguistic Code) (3-7 mistakes/page)	(Linguistic Code) (8-10 mistakes/ page)	(Linguistic Code) (>10 mistakes/ page)
- 0	**- 0,5 - 2.5**	**- 2.5 - 3.5**	**- 4**