Programming Variation Assignment

Intake 2027-2028



CREATING MEANINGFUL EXPERIENCES

Academy for AI, Games, Media

31 August 2026

Breda University OF APPLIED SCIENCES

Programming Intake Assignment 2027-2028

Theme: CATCH

Creative Media and Game Technologies - Programming Track

What this assignment is about

We are not looking for a commercial-quality game or perfect C++ code. We want evidence that you are willing to learn, invest effort, solve problems, build a small graphical game, and understand the code you submit.

At a glance

Language	C++
Game	Small graphical game based on CATCH
Tools	Lightweight libraries/frameworks allowed; complete game engines are not
Main evidence	Effort, C++ fundamentals, understanding, problem solving, ownership
Selection	Your assignment and portfolio are reviewed first; an interview is by invitation
Interview	If invited: approximately 30 minutes, with an important focus on discussing your own code

1. Purpose of the Assignment

This assignment is part of the admission process for the Programming track of the Creative Media and Game Technologies programme at Breda University of Applied Sciences.
Many applicants are completely new to C++ or game development. That is fine. The purpose of this assignment is not to find out whether you are already an experienced programmer.
Instead, we want to see whether you are willing and able to:

- invest time and effort into learning programming;
- learn and apply the fundamentals of C++;
- solve problems and learn from mistakes;
- create a small but complete graphical game;
- understand the code you submit;
- explain the decisions you made while developing your game.

Important

A small game that you understand well is much better than a large or technically impressive project that you cannot explain.

2. Getting Started

If this is your first experience with C++ and game programming, we strongly recommend following this tutorial series:
https://www.3dgep.com/cpp-fast-track-1-getting-started
Work through the tutorial step by step rather than only copying the final result. Try to understand what the code does, experiment with it, make mistakes, and solve problems as you go.
Learning programming takes time. That is part of what this assignment is designed to assess.
The PowerPoint template that accompanies this assignment becomes available through My Online Application as you follow the application procedure.
All submitted materials, including the PowerPoint presentation and readme.txt, must be in English.

3. The Assignment

Create a small graphical game written in C++. Your game must be based on the theme:

CATCH

You may interpret the theme in different ways. For example, your game could involve:

- catching falling or moving objects;
- catching or capturing another character;
- avoiding being caught;
- catching projectiles, signals, creatures, or resources;
- using a catch-and-release mechanic;
- a rule or system that interprets “catch” in another clear gameplay-related way.

A simple interpretation is completely acceptable. The theme is intended to give you a creative starting point. We are primarily assessing your programming and learning process, not the originality of your interpretation of the theme.

4. Keep the Scope Small

You do not need to create a large game. Games at approximately the scale of Breakout, Space Invaders, Asteroids, a simple top-down game, or a simple platform game are sufficient.
Your game should contain enough interaction to demonstrate that you have learned and applied basic programming concepts.

Keep this priority in mind

Completing, testing, understanding, and being able to explain a small project is more important than adding many features.

5. Minimum Requirements

C++

The game must be programmed in C++.

Graphical Gameplay

Your game must display graphical content and contain graphical gameplay. A text-only or console-only game is not sufficient.
You may use sprites, images, simple geometric shapes, 2D or 3D models, text, sound, and other appropriate assets. The quality of your artwork is not an important assessment criterion.

No Complete Game Engine

You may use libraries and frameworks that help you create graphics, audio, input, windows, and other basic functionality. For example:

- the provided tutorial template from the 3dgep website: https://www.3dgep.com/cpp-fast-track-2-template/
- SFML (a modern, easy to use library that is highly recommended);
- raylib;
- similar lightweight libraries.

You may not use a complete game engine such as Unity, Unreal Engine, Godot, GameMaker, or another engine that provides most of the game structure and behaviour for you.
The purpose of this requirement is to make sure that you have to program the important behaviour of your game yourself.

6. Programming Concepts We Expect to See

This assignment should demonstrate that you have started learning the fundamentals of C++ programming. We expect your project to meaningfully use concepts such as the following.

Variables and fundamental data types

Use values such as integers, floating-point numbers, and booleans to represent meaningful game state.

```
int score;
float playerSpeed;
bool gameOver;
```

You should understand what the values represent and why an appropriate type was selected.

Conditions

Use conditions such as if and else to make decisions. Examples include checking for collisions, reacting to input, testing win/lose conditions, or checking whether an object has reached a boundary.

Iteration

Use loops such as for or while where appropriate. Examples include updating multiple objects, drawing projectiles, checking several objects for collisions, or repeating an operation.

Functions

Divide parts of your program into functions where this makes sense. You should be able to explain what a function does, what information it receives or returns, and where it is called.

```
UpdatePlayer();
CheckCollisions();
SpawnObject();
DrawGame();
```

Classes or structs

Your program should contain at least one meaningful programmer-defined type using a class or struct, for example Player, Enemy, Projectile, CatchableObject, or another type that represents something meaningful in your game.
Do not create classes simply to satisfy this requirement.

Collections

Where appropriate, you may use an array, std::vector, or another suitable collection to store multiple objects such as enemies, projectiles, or catchable objects. If you use one, you should understand why it is useful and how you access the objects stored inside it.

Game state and interaction

Your game should contain behaviour that changes while the game is running, such as movement, spawning, score changes, collisions, health, timers, or switching between game states.

Time-based behaviour

If your game contains movement or other behaviour that depends on time, we encourage you to investigate concepts such as delta time. If you use delta time, you should be able to explain what it represents, where you obtain it, where you use it, and why it is useful.

7. Code Structure

We do not expect professional-quality software. We do expect you to make an effort to keep your program understandable.
Try to:

- give variables and functions meaningful names;
- divide larger problems into smaller parts;
- avoid unnecessarily repeating the same code;
- keep related functionality together;
- use comments where they help explain something that is not obvious.

Do not add comments simply to describe every individual line. Comments are more useful when they explain why something was implemented in a particular way.

Most important

You must understand the code you submit.

8. Applicants With Previous Programming Experience

Some applicants will already have programming or game-development experience. If that applies to you, you are encouraged to go beyond the minimum requirements and show us what you can do.
You might choose to explore subjects such as:

- more sophisticated gameplay systems;
- interesting algorithms;
- more complex enemy behaviour;
- procedural generation;
- pathfinding;
- data-driven systems;
- advanced mathematics;
- graphics techniques;
- interesting uses of the C++ standard library;
- more advanced code organisation;
- tools you created yourself.

These are examples, not additional requirements.

Complete your project

A smaller finished project is better than a very ambitious project that is incomplete or does not work.

Understand what you submit

You may be asked about any important part of your project during the interview. Technical complexity by itself does not guarantee a stronger assessment. We are interested in what you have learned, what you have created, and what you understand.

9. Using Tutorials, Documentation, AI, and Other Help

Programming is rarely done without external resources. You may use the provided tutorial, books, documentation, websites, example code, programming forums, videos, AI tools, and help from other people.

Using AI

AI tools such as ChatGPT, Claude, GitHub Copilot, or similar tools may be used while learning and working on this assignment.
For example, you may use AI to:

- ask what a compiler error means;
- ask for an explanation of a C++ concept;
- understand how a library works;
- ask for help solving a particular programming problem;
- get feedback on your code;
- help you find a bug;
- generate or improve part of an implementation.

However, you remain responsible for everything you submit. Do not simply submit code because an AI generated it and it appears to work.
You must make an effort to understand what the code does, how the important parts work, how the different parts connect, and why particular solutions were used.

AI use is allowed; understanding is required

If you used AI to help create part of your program, that is acceptable. If you cannot explain the resulting code, that part of the submission does not demonstrate your programming understanding.

You do not need to provide a complete history of every AI prompt you used. Instead, be open about how you used AI and other resources and be prepared to discuss what you learned from them.

10. External Code, Libraries, and Assets

You may use existing images, textures, 3D models, audio, fonts, libraries, tutorials, and example code. You do not receive additional programming credit for creating artwork yourself.
Clearly acknowledge important external resources that contributed to your project. List the main learning resources you used on the PowerPoint Summary slide. Use your readme.txt or other project documentation to acknowledge external assets, libraries, and substantial external code where appropriate. Your submission should represent your own development and learning process, even when you have used external resources to help you.

11. Using the PowerPoint Template

The PowerPoint template is part of your submission. Follow the instructions on the slides and keep the provided structure and formatting. Do not spend time redesigning the presentation. All content must be in English.

Education information

The template asks for your current education level. If you have studied programming or game development at higher or college level, state the institution, course, and diploma or degree awarded. Dutch applicants should also answer the Technasium question where applicable.

Assignment summary

On the Programming Intake Assignment - Summary slide, briefly state:

- whether you already had programming experience and, if so, whether you used C++ before;
- a rough estimate of how much time you spent on the assignment;
- the main learning resources you used, including tutorials, documentation, books, websites, AI tools, or other help where relevant.

Game description and media

Provide a short description of your game and add a screenshot or video clip of the game in action on the slides provided for this purpose.

Portfolio, skills, and knowledge

The template also asks you to provide relevant portfolio work, skills, knowledge, and activities that help show your current experience and preparation for studying game development. In the portfolio section, you may insert up to 7 slides. Images are allowed, with a maximum of 200 words per slide. Do not add extra slides elsewhere unless the template explicitly allows them.

Development process and interview preparation

The PowerPoint is not intended to be a detailed development diary. You do not need to add separate slides about every bug, iteration, or thing you learned unless the template specifically asks for it. If you are invited to an interview, however, be prepared to discuss what you learned, problems you encountered, decisions you made, and how your game changed during development.

12. Assessment and Interview Process

First stage: assignment and portfolio review

Before any interview, we review only the submitted programming assignment and the portfolio evidence provided in your PowerPoint. Not every applicant who submits the assignment is automatically invited to an interview.

Applicants whose submitted work meets the level needed to progress in the admissions process may be invited to an interview. At this stage, the completed assignment itself is important: we look for a serious attempt, evidence of effort and learning, use of appropriate programming fundamentals, and a submission that is complete enough for us to review and run.

This is an admissions assessment rather than a course assignment. The internal rubric and assessment score are not shared with applicants.

What we look for

The overview below describes the kinds of evidence that may contribute to our assessment. It is not a checklist of items that we always assess separately, and different projects may demonstrate these qualities in different ways. We do not necessarily ask every applicant the same questions. One part of your project or one interview answer may provide evidence in several areas at the same time.

Area	Examples of evidence we may consider
Effort and engagement	Evidence that you seriously engaged with the assignment: experimenting, solving problems, testing, improving, and learning.
C++ fundamentals	Meaningful use of basic programming concepts such as variables, types, conditions, loops, functions, classes/structs, and collections where appropriate.
Understanding	Evidence that you understand how the important parts of your own program work.
Problem solving	Evidence of how you approached problems, investigated failures, and learned from mistakes.
Ownership of your work	Evidence that the project represents work you can navigate, discuss, and explain, including code created with external help.
Completion	The submission can be opened, compiled, run, and played as a complete small project.
Curiosity and independent learning	Evidence that you investigated, experimented with, or learned something beyond simply following instructions.

13. The Interview

If you are invited, the interview takes approximately 30 minutes. An important focus of the interview is a discussion of the C++ game you submitted. We use your own project as the starting point for questions about programming, your development process, and what you learned. The interview does not include a live-coding exercise.

We may ask questions about your own code, for example:

- Show us where you store the player's position.
- Show us where player input is handled.
- How do you move the player?
- Show us how you detect that the player caught an object.
- What does this if statement check?
- Why did you use a loop here?
- Where are your catchable objects, enemies, or projectiles stored?
- What is this class responsible for?
- Where is this function called?
- What happens when this variable changes?
- What was a difficult bug you encountered?
- Which part of the project taught you the most?
- Which part would you improve if you continued working on the game?

If you use delta time, we may ask you to explain how and why. If your project contains more advanced programming techniques, we may ask you about those as well.

The interview also gives us evidence about other aspects of your application, such as your motivation, reflection, and how clearly you can discuss your work. These aspects often overlap: for example, explaining a piece of code can show both programming understanding and your ability to communicate your reasoning.

14. Development Environment

Your submitted project must compile using Visual Studio Community:
https://visualstudio.microsoft.com/vs/community/

Test your project before submitting it. Where applicable, test both the Debug and Release configurations.
A configuration or packaging problem is not the main focus of our assessment, but we do need to be able to open, compile, and run your project.

15. Game Instructions

Include a readme.txt file with your project. It should explain:

- the name of the game;
- how to play;
- which keyboard keys, mouse buttons, or controller buttons are used;
- any special instructions needed to run or play the game.

Keep it simple and clear.

16. What to Submit

1. Complete project ZIP file

Your ZIP file should contain everything required to open, compile, and run your project, including:

- all .cpp files;
- all .h files;
- Visual Studio solution/project files;
- all required game assets;
- all required external libraries and dependencies;
- readme.txt.

2. PowerPoint presentation

Complete the PowerPoint presentation using the template provided through the application portal. Follow the prompts on the slides. The template asks for education information, assignment summary information, a game description, media showing your game in action, and relevant portfolio, skills, and knowledge evidence. Upload the PowerPoint together with your project.

17. Important: Include Your Dependencies

If you use SFML, raylib, or another external library, make sure that the files required to compile and run your project are included. We should not need to download additional libraries before we can build your game.
A common submission problem is that a project works on the applicant's computer but not on another computer because required files or library paths are missing.
Where possible, keep external dependencies inside your project directory and configure Visual Studio to use paths relative to your project rather than paths that only exist on your own computer.

```
MyGame/
    src/
    assets/
    libs/
        SFML/
```

18. Test Your Submission

Before uploading your project:

1. Create your ZIP file.
2. Move or copy it somewhere else.
3. Extract it into a completely new folder.
4. Open the extracted project.
5. Compile it.
6. Run it.
7. Check that the assets load correctly.
8. Check that the game can actually be played.

If possible, also test it on another computer. This is one of the best ways to discover missing files before submitting.

19. Resources for Beginners

Online tutorial

https://www.3dgep.com/cpp-fast-track-1-getting-started
We strongly recommend this tutorial if you are new to C++.

Recommended book

Beginning C++ Through Game Programming, 4th Edition - Michael Dawson
ISBN: 9781305109919
https://books.google.nl/books?id=pd0OBAAAQBAJ
The book provides an introduction to many important C++ programming concepts, although it does not focus on graphical programming.

20. What a Successful Submission Looks Like

A successful beginner submission can be a relatively simple game. For example, imagine a game where the player moves a catcher at the bottom of the screen while objects fall from above.
The game might contain:

- player movement;
- multiple falling objects;
- collision or catch detection;
- a score;
- missed objects or lives;
- a game-over and restart condition.

The code does not need to be sophisticated. What matters is that you can explain how the player's position is represented, how input changes that position, how objects are stored and updated, how catches are detected, how the score changes, why loops and conditions are used, what the main classes or structs represent, and what you learned while implementing the game.

21. Submission Checklist

Use this checklist to check the practical submission requirements before uploading. The assessment areas described earlier are considered holistically; they are not a separate checkbox list.

Game requirements

☐ My game is based on the theme CATCH.
☐ My game is written in C++.
☐ My game has graphical gameplay.
☐ I did not use a complete game engine.
☐ My project includes meaningful use of the C++ programming fundamentals described in this assignment.
☐ My game represents a complete small project that can be run and played.

Project package

☐ All .cpp and .h files are included.
☐ All assets are included.
☐ All required libraries and dependencies are included.
☐ Visual Studio solution/project files are included.
☐ readme.txt is included.
☐ The project is contained in a single ZIP file.

Documentation and attribution

☐ My PowerPoint presentation is complete and follows the prompts and slide limits in the provided template.
☐ I have listed my main learning resources in the PowerPoint and acknowledged external assets, libraries, and substantial external code in my readme.txt or project documentation where appropriate.
☐ My readme.txt clearly explains how to run and play the game.

Testing

☐ I extracted the ZIP into a new folder and compiled it successfully.
☐ The game runs correctly and assets/dependencies are not missing.
☐ I tested Debug and Release configurations where applicable.

Final Advice

Do not worry if you are new to programming. We are not looking for applicants who already know everything we will teach during the programme. We are looking for evidence that you are willing to learn, experiment, solve problems, and invest effort into becoming a programmer.
If you already have programming experience, you are welcome to challenge yourself and impress us with what you can create.

In both cases, the same principle applies
Create something you can finish, understand, and explain.



Games



Leisure & Events



Tourism



Media



Data Science & AI



Hotel



Logistics



Built Environment



Facility

Mgr. Hopmansstraat 2
4817 JS Breda

P.O. Box 3917
4800 DX Breda
The Netherlands

PHONE
+31 76 533 22 03
E-MAIL
communications@buas.nl
WEBSITE
www.BUas.nl

CREATING MEANINGFUL EXPERIENCES

Breda University
OF APPLIED SCIENCES