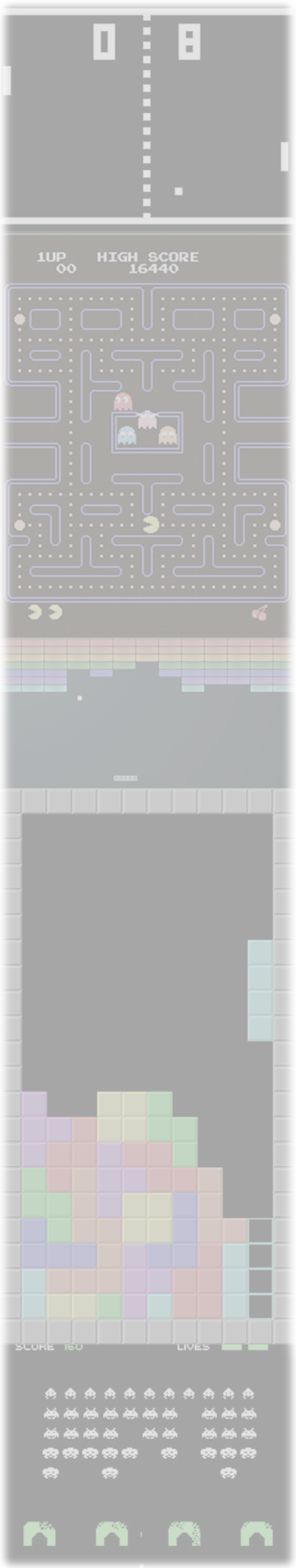


From Code to Game

Beginning 2D Game Programming with SDL3

Timeline

- 19 November 2025
  - Online training: Using the basic functionalities of the SDL3 library
- 22 November 2025
  - CodeRun @TUCN

TUCN Trainer

- Adrian SABOU – Senior Lecturer, Computer Graphics and Interactive Systems (CGIS) Group, Department of Computer Science, Faculty of Automation and Computer Science, Technical University of Cluj-Napoca (TUCN)

Technical Specifications

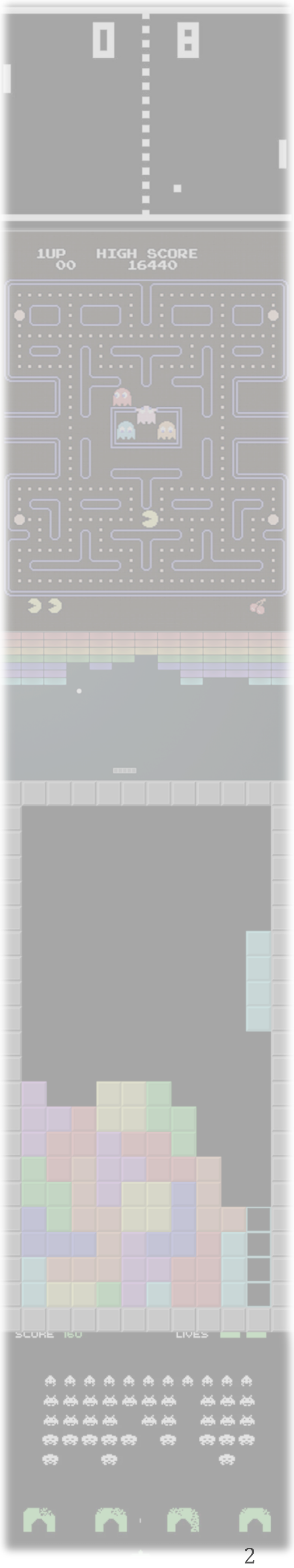
- Required operating system: any operating system for which the SDL3 library has support (Microsoft Windows, Linux, macOS)
- Programming languages: C/C++
- Required IDEs: any C/C++ capable IDE (e.g. Microsoft Visual Studio, CLion, Xcode, etc.)
- Auxiliary libraries: The SDL3 library for programming 2D Games (https://www.libsdl.org), The OpenGL mathematics library (GLM) (https://glm.g-truc.net/0.9.9/)
- Prerequisites: Basic C/C++ programming skills
- Pre-work: Configuring your IDEs to compile and run a "Hello SDL3 Window" Application (link to code). You can use any tutorials available online for this task. **The code provided must compile as it is, without any alterations to it!** Pre-configured projects are available for Microsoft Visual Studio 2022 (Microsoft Windows), CLion 2024.2.2 (Microsoft Windows, both for the VS2022 and the bundled MinGW Toolchains), CLion 2024.2.2 (macOS) (link to projects). For CLion on macOS, please consult the following document for the instructions to follow after you download the project (link to document).

Training Outline

- Trainer introduction
- SDL3 library overview
- Understanding a basic SDL3 application (Hello SDL3 Window!)
- Basic user interaction in SDL3
- Drawing basic shapes in SDL3
- Working with time in SDL3
- Putting it all together in a simple mini game

A Blast From the Past

Implementing a Block Breaker Arcade Game

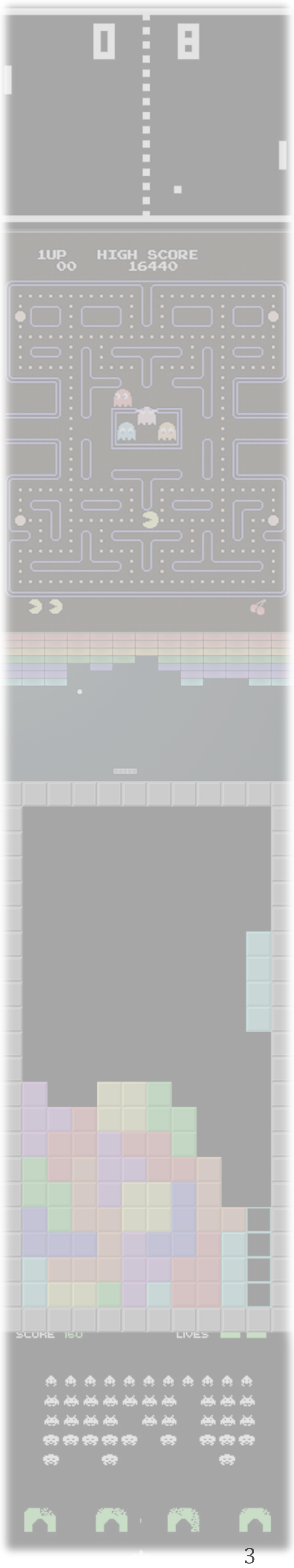


Context

- In 1976, Atari released one of the most famous block braking arcade games, Breakout. Gameplay involved the player using a paddle to bounce a ball all around the screen. The game's objective was to score points by breaking all the blocks using the ball (video).
- In this context, your task is to implement a very simple block breaker arcade game in SDL3.
- You will start from the from the SDL3 application **template** provided here. **Using the SDL3 library and the code template is mandatory and implementations that do not use them will not be considered for scoring.**

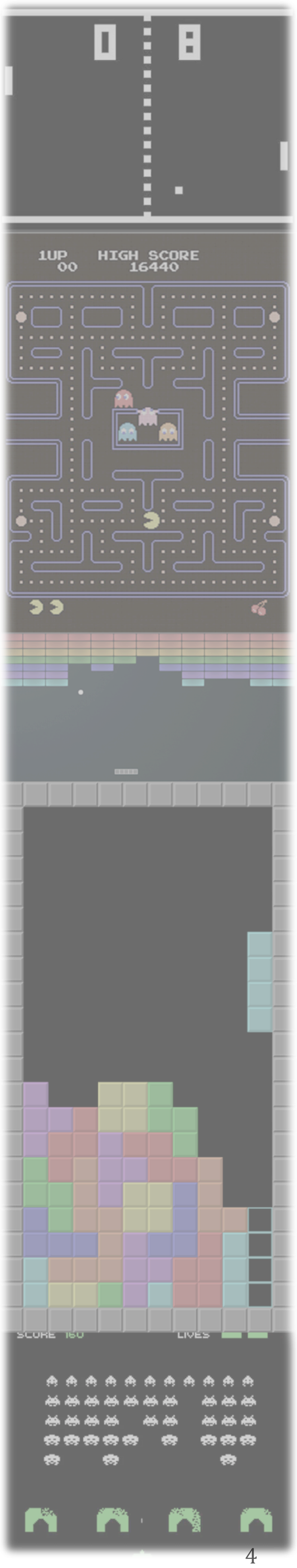
Easy Task

- Your game will be similar to the following demo (demo link)
- The goal of the game is to launch and catch a ball using a paddle in order to break all bricks on screen
- Your SDL3 window will be 800 pixels wide and 800 pixels tall
- There will be 17 bricks in one row in the upper part of the window, 100 pixels from the upper window edge
- Bricks will be green, 20 pixels tall, and their width must be computed in such a way as to fit the entire window width with bricks
- The paddle will be blue, 100 pixels wide and 25 pixels tall, and will be placed at the bottom of the window
- The ball will be orange, 10 pixels in radius and initially placed on top of the paddle, its center aligned horizontally with the paddle's center
- The user can move the paddle left and right using the arrow keys, without allowing the paddle to exit the window
- While the ball is on the paddle, it must move along with it
- When the user presses the "Space" key, the ball will be launched upwards with constant speed
- The ball will bounce whenever it hits the left, right or upper window edge
- When the ball hits a brick, the brick will be destroyed (it will disappear) and the ball will bounce back downwards on a random direction, making an angle between -80 and 80 degrees with the down direction
- When the ball hits the paddle, it will stick to it until the "Space" key is used to launch it again upwards
- When all bricks have been destroyed, the game ends (all animations stop), the paddle turns green, and the ball is no longer visible
- If the ball reaches the window's lower edge, the game ends (all animations stop), and the paddle turns red
- After the game ends, it can be reset to its original state using the "R" key
- The game cannot bet reset before it ends



Hard Task

- Your game will be similar to the following demo (demo link)
- The goal of the game is to launch and bounce a ball using a paddle in order to break all bricks on screen
- Your SDL3 window will be 800 pixels wide and 800 pixels tall
- There will be 3 rows of 17 bricks each in the upper part of the window, the top row 100 pixels from the upper window edge
- Bricks have 1, 2 or 3 lives; bricks with 3 lives are green, bricks with 2 lives are yellow and bricks with one life are red
- Bricks in the top row have 1 life, bricks in the middle row have 2 lives and bricks in the lower row have 3 lives
- Bricks will be 20 pixels tall, and their width must be computed in such a way as to fit the entire window width with bricks
- The paddle will be blue, 100 pixels wide and 25 pixels tall, and will be placed at the bottom of the window
- The ball will be orange, 10 pixels in radius and initially placed on top of the paddle, its center aligned horizontally with the paddle's center
- The user can move the paddle left and right using the arrow keys, without allowing the paddle to exit the window
- While the ball is on the paddle, it must move along with it
- When the user presses the "Space" key, the ball will be launched upwards with constant speed
- The ball will bounce whenever it hits the left, right or upper window edge
- When the ball hits a brick, the brick will lose a life and the ball will bounce of the brick
- When a brick loses all its lives, it will be destroyed (it will disappear)
- When the ball hits the top of the paddle, it will bounce back upwards, on a direction that makes an angle between -80 and 80 degrees with the up direction, based on the point of contact (ex. if the ball hits the leftmost point of the paddle it will bounce back on a direction that makes -80 degrees with the up direction; if the ball hits the rightmost point of the paddle it will bounce back on a direction that makes 80 degrees with the up direction; if the ball hits the middle point of the paddle's upper edge, it will bounce back on the up direction)
- When all bricks have been destroyed, the game ends (all animations stop), the paddle turns green, and the ball is no longer visible
- If the ball reaches the window's lower edge, the game ends (all animations stop), and the paddle turns red
- After the game ends, it can be reset to its original state using the "R" key
- The game cannot bet reset before it ends



- **Important notes! (For both tasks)**
  - **All objects' placements are relative to their center**
  - **All positional constraints must be fulfilled even if the size of the window changes (so no hard coding positions based on the 800x800 window size)**
  - **You must design your game to be able to run <u>indefinitely</u>**

Easy Task Scoring

- Attendance: 10 points
- Correct visual implementation of all game elements (bricks, paddle, ball): 10 points
- Correct implementation of the paddle's constrained movement: 10 points
- Correct implementation of ball launching, ball catching and ball bouncing of walls: 20 points
- Correct detection of collisions between ball and bricks: 10 points
- Correct implementation of ball bouncing of bricks: 20 points
- Correct implementation of game winning, losing and resetting scenarios: 10 points
- Constant, fluid and correct movement speed of all game objects: 10 points
- Time bonus: maximum 10 points (0.5 points / minute, for a maximum of 20 minutes); **Note:** to receive the time bonus, the task needs to be at least 50% solved correctly

Hard Task Scoring

- Attendance: 20 points
- Correct visual implementation of all game elements (bricks, paddle, ball): 20 points
- Correct implementation of the paddle's constrained movement: 10 points
- Correct implementation of ball launching and ball bouncing of walls: 20 points
- Correct implementation of ball bouncing of the paddle: 40 points
- Correct detection of collisions between ball and bricks: 10 points
- Correct implementation of ball bouncing of bricks: 40 points
- Correct implementation of bricks' lives system: 20 points
- Correct implementation of game winning, losing and resetting scenarios: 10 points
- Constant, fluid and correct movement speed of all game objects: 10 points
- Time bonus: maximum 20 points (1 point / minute, for a maximum of 20 minutes); **Note:** to receive the time bonus, the task needs to be at least 50% solved correctly