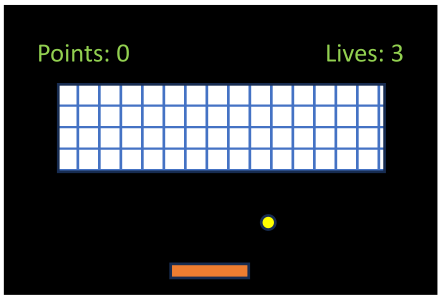
Programming and Data Structures 2025
Exercise 5
Breakout Game in Python

Overview

The purpose of this laboratory exercise is to learn more about Python, classes, objects, collections, composition, inheritance, polymorphism, delegation, modularity, graphics, event handling and real-time programming.

Assignment

Construct a Breakout-game, a simple brick-breaking game a la the Atari 2600 version. A Breakout-game consists of a gameboard that has a number of bricks in the upper part, a user-controlled bat at the bottom and a bouncing ball that moves around. The objective is to bounce the ball against the bricks to destroy them, and stopping the ball from falling below the bottom of the gameboard with the bat.



Game Concepts

The following concepts **must be** implemented in the game:

- **Playability**, the game must be playable, but not necessarily fun ☺.
- **Walls**, either visible or just being the limits of the game board, left, right and top. There should be no wall at the bottom.
- **Sprite**, a superclass for visual objects on game board.
- **Ball**, moving on its own accord, bouncing off walls, bricks and bat.
- **Brick(s)**, at least two kinds that require different number of hits to be destroyed and also gives different points when destroyed. Colors should indicate which type of brick it is.
- **Bat**, controllable by the user by pressing ←→.
- **Points**, should be displayed in the gameboard.
- **Lives**, should be displayed in the gameboard.
- **Game status**, running / game over.

Gameplay: The game should start by displaying a number of bricks in at least three rows with at least 8 bricks per row, a bat at the bottom and a ball. Points should be 0 and lives at least 3, both should also be displayed in the gameboard. The ball should either start moving directly or start when the user presses a key on the keyboard. As long as there are still bricks left and the player has lives left the game should continue. The game ends either when all bricks have been destroyed or when all lives are lost. When finished, the game should display a message indicating whether the game was won or lost.

The following concepts **might be** implemented if you want to and find the time and motivation, not obligatory (but fun):

- Restarting of game
- High score list
- Animated sprites
- Sound effects
- Background music
- Levels
- Powerups (Some bricks "contain" powerups that are either immediately gotten, or causes a symbol to fall from the brick that must be hit with the bat.)
    - Extra points (just adds points to the score)
    - Extra lives (increases the player's lives)
    - Multi-ball (more than one ball active at the same time)
    - Speed-up (ball increases speed)
    - Laser ball (does not bounce off bricks but goes through)
    - Sticky bat (ball sticks to bat and can be fired by pressing a key)
    - Long bat (bat becomes longer)
    - Fast bat (bat becomes faster to move)
    - Slow bat (bat becomes slower to move)
    - Powerup of your invention, anything goes

Work flow

Analysis – Think about what properties and operations each class should implement and what relations it should have to other classes.

Design – Make a class diagram based on the analysis that shows classes with their properties, operations and relations to other classes. Use the OO techniques taught in the module (encapsulation, composition, inheritance, interfaces, polymorphism). Draw a sketch of how you want the game board to look.

Implementation and Test – Start by generating an empty game board, add a ball that moves, make it bounce against walls, add a bat and make the ball bounce off the bat, add simple bricks that disappear when hit. Add points and lives.

Demonstration

The assignment must be demonstrated to a lab supervisor (TA) or teacher.

Report

A short report is required, max 4 pages including introduction, overview (class diagram), interesting details, spent time, and a summary of lessons learned.