Emergent Architecture Design

Most recently modified on: 30-4-2015

(Note the word “currently” has many occurrences. This can be read as a synonym of “at the time of writing this document”)

1. Introduction

1.1. Design goals

We will try to keep system structure as simple as possible (without having a negative effect the end product). Since we have a very specific setup which imposes many constraints on the possible locations where the game can be realised. Our hardware setup needs to be able to perform correctly for a real queue, without being an obstacle to the environment. Furthermore the player input via camera detection needs to be able to run smoothly in real time for the product to be playable.

2. Software architecture views

2.1. Subsystem decomposition (sub-systems and dependencies between them)

Currently we foreshadow our main modules to be:

- Player Input system
    - Hardware:
        - Camera
        - Computer
    - Software:
        - Camera image describes silhouettes (background - foreground)
        - Algorithm detects important points which will be interpreted as the player controls and used in the gameplay
    - Makes use of data from:
        - No other sub-system
- Game controller:
    - Hardware:
        - Computer
    - Software:
        - Process player input received from the “player input system”
        - Position the object, slope, targets and obstacles
        - Check when the object reaches the target
        - Handle event object-obstacle collision
        - Keep track of combo count
        - Decide when to display powerups and when to trigger bonus levels
    - Makes use of data from:
        - Player input system
        - Game physics system

- Game physics system:
    - Hardware:
        - Computer
    - Software:
        - Set gravity, friction and other physics rules
        - Compute the line / slope using the game controller data
        - Compute how the object moves along the line
    - Makes use of data from:
        - Game controller
- Display system:
    - Hardware:
        - Screen/Projector
        - Computer
    - Software:
        - Output the positions of the gameplay elements to the display
        - Output the GUI to the screen
        - Draw the graphics (textures, particle systems…)
    - Makes use of data from:
        - Game controller

2.2. Hardware/software mapping (mapping of sub-systems to processes and computers, communication between computers)

Currently we are planning on using only one computer. It will only be running our game. Since our game will be written in Java, the computer needs a JVM to execute our game.

2.3. Persistent data management (file/database, database design)

Currently our vision is such that our game will not make use of a database, so no persistent data management will be necessary. Only if we incorporate high scores we will have them written to a file.

2.4. Concurrency (processes, shared resources, communication between processes, deadlocks prevention)

Our game will not consist of multiple processes, so no interprocess communication will have to be implemented. Neither on the machine running our game or on other machines via the Internet, as our game will not make use of the Internet.

2.5. Glossary

No need for a glossary at time of writing.