Multiplayer Platform Game

Martin Petterson
Oskar Kvist
Christoffer Ekeroth
Misael Berrios Salas

January 28, 2008

Contents

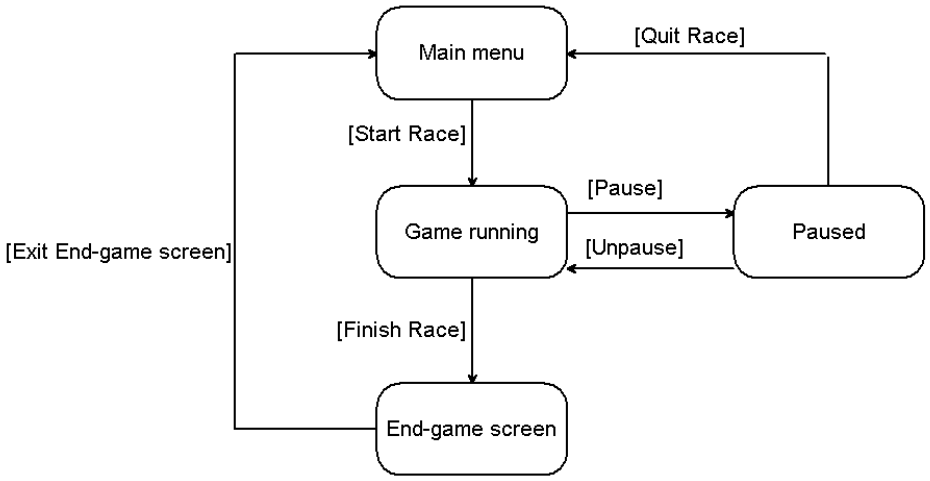
1 Introduction

2 System Overview

2.1 General Description

2.2 Overall Architecture Description

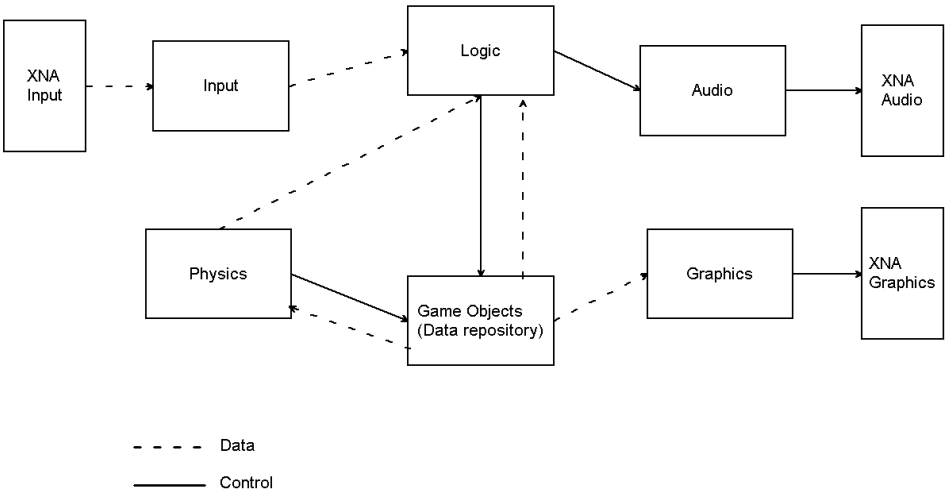
The following is a state transition diagram showing the states the game can be in as well as how the game can change state.



The game can be in any of four states at any given time. These states are:

- *Main Menu*—This is the initial menu presented to the players upon startup and before the start of each race. In order to start a race players select characters and a game stage.
- *Game Running*—In this state the players play the actual game (sometimes referred to as a "race").
- *Paused*—In this state the race is paused and the players are presented with a menu.
- *End-Game Screen*—In this state the players are presented with the positions they finished in and their times.

The following diagram shows the subsystems the game is composed of as well as data and control flow between the subsystems.



- The *Input subsystem* is responsible for accepting input from the players.
- The *Logic subsystem* is responsible for enforcing game rules on the world objects as well as handling menu navigation.
- The *Audio subsystem* is responsible for playback of music and sound effects.
- The *Physics subsystem* is responsible for calculating the positions and velocities for game objects as well as detecting collisions between them.
- The *Graphics subsystem* is responsible for drawing to the screen.
- The *Game Objects subsystem* is a data repository containing information about game objects such as players, traps, monsters, platforms, etc.

The game runs in a continous loop, in all states of the game. During every iteration of the loop the following things happen:

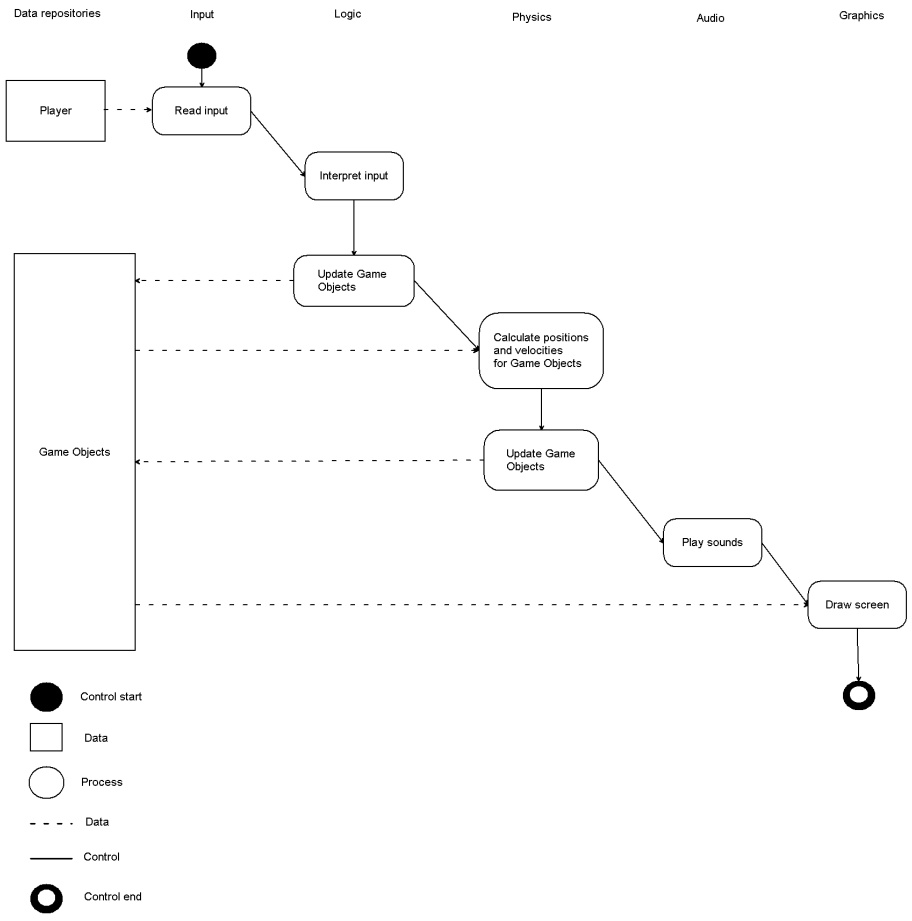
1. The Input subsystem checks for player input. Player input is sent to the Logic subsystem.
2. The Logic subsystem interprets the player input, taking different actions depending on what state the game is in.

3. If the game is in the Game Running state, the Physics subsystem is used to calculate positions and velocities for all game objects and to detect collisions between them.

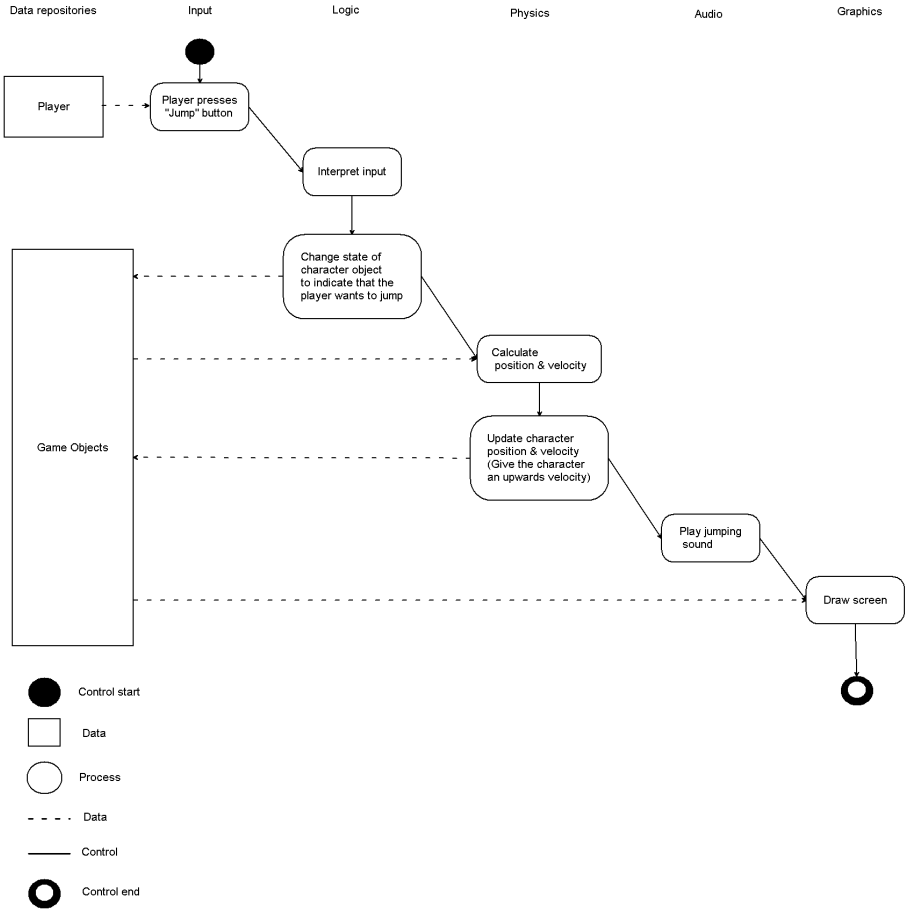
4. After the Logic subsystem has updated the game, the Audio subsystem plays the appropriate sound effects and the Graphics subsystem updates the graphics on the screen.

2.3 Detailed Architecture

The following diagram shows data and control flow during an iteration of the game loop in the general case.



The following diagram shows data and control flow during an iteration of the game loop in the case of a player jumping.


Data repositories
Input
Logic
Physics
Audio
Graphics
Player
Player presses "Jump" button
Interpret input
Change state of character object to indicate that the player wants to jump
Game Objects
Calculate position & velocity
Update character position & velocity (Give the character an upwards velocity)
Play jumping sound
Draw screen
Control start
Data
Process
Data
Control
Control end