Platitude: Requirements

CS121 - Software Development | Fall 2010

Harvey Mudd College

Project team: Philip Aelion-Moss, Christopher Beavers, David Cook, Sarah Ferraro, Aaron Gable, Leif Gaebler, Jacob Heller, My Ho, Sean Laguna, Kathryn Lingel, Camille Marvin, Stephanos Matsumoto, Beatrice Metitiri, Emily Myers-Stanhope, Oliver Ortlieb, Thea Osinski, Richard Porczak, Kevin Riley, Max Strater, Rahul Swaminathan, Russel Transue, Heather Williams, Lilian de Greef, Alexandre François (pm).

Table of Contents

External Interface Requirements

This section describes user and system requirements that affect the operation of the game within a given system or within a network context.

User Interface

Hardware Interfaces

The target platform for the game clients is the Apple MacBook? Pro. For convenience, the game server should also run on that platform.

Ideally, a game server and player client should be able to run simultaneously on the same machine for a stand-alone, single-player configuration.

Software Interfaces

The system makes use of established libraries such as OpenGL for graphics, GLUT for window and interaction management, etc.

The system is implemented using the MFSM architectural middleware; and existing relevant functional modules whenever available.

Communication Interfaces

The system implements a client-server architecture which involves one game server and several player and spectator clients.

=> *to confirm:* The system uses BSD sockets to support communications between individual game components.

Functional Requirements

Requirements List

(0) The software shall support multiplayer.

(1) The software shall support interaction with a server and clients.

(2) The software shall support text-based communication between players in the lobby.

(3) The software shall, upon request, divide teams fairly according to player's records.

(4) The software shall allow user to select a mode of gameplay.

(5) The software shall allow builder player to place obstacles to obstruct platformer player.

(6) The software shall allow builder player to upgrade obstacles.

(7) The software shall allow input from keyboard and mouse.

(8) The software shall allow platformer player to move his character (left, right, and jump).

(9) The software shall show game statistics during gameplay, i.e. resources, health, and time.

(10) The software shall monitor resources available to the builder.

(11) The software shall show the game map to users.

(12) The software shall allow builder player to zoom in and out.

(13) The software shall move map view with platformer's location.

(14) The software shall support a physics engine.

(15) The software shall provide game instructions upon request.

(16) The software shall allow user to quit anytime.

(17) The software shall enforce rules for fair gameplay.

(18) The software shall support music.

(19) Software shall support volume and mute controls and hotkeys at any point during gameplay.

(20) Software shall have two musical options: loops developed specifically for and included with the game (1), and user-uploaded music (2).

(21) Within option 1, software shall play layers of loops in accordance to the on-screen game status (items present and player health).

(22) Within option 2, software shall allow for users to store and play songs as playlists, or play classified music at appropriate game points.

(23) The software shall have sound effects.

(24) The software shall support a database of player profiles containing player's name, win/lost record, preference for mode of gameplay, etc.

(25) The software shall support builder AI. (see 29)

(26) The software shall have AI replace a user when a user quit prematurely.

(27) The software shall support 2D graphics.

(28) The software shall show home screen (lobby) before and after games.

(29) The software shall support platformer AI.

Components View

Networking

(0) The software shall support multiplayer.

(1) The software shall support interaction with a server and clients.

(2) The software shall support text-based communication between players in the lobby.

Game Play

(3) The software shall, upon request, divide teams fairly according to player's records.

(4) The software shall allow user to select a mode of gameplay.

(5) The software shall allow builder player to place obstacles to obstruct platformer player.

(6) The software shall allow builder player to upgrade obstacles.

(7) The software shall allow input from keyboard and mouse.

(8) The software shall allow platformer player to move (left, right, jump).

(9) The software shall show game statistics during gameplay, i.e. resources, health, and time.

(10) The software shall monitor resources available to the builder.

(11) The software shall show the game map to users.

(12) The software shall allow builder player to zoom in and out.

(13) The software shall move map view with platformer's location.

(14) The software shall support a physics engine.

(15) The software shall provide game instructions upon request.

(16) The software shall allow user to quit anytime.

(17) The software shall enforce rules for fair gameplay.

Music

(18) The software shall support music.

(19) Software shall support volume and mute controls and hotkeys at any point during gameplay.

(20) Software shall have two musical options: loops developed specifically for and included with the game (1), and user-uploaded music (2).

(21) Within option 1, software shall play layers of loops in accordance to the on-screen game status (items present and player health).

(22) Within option 2, software shall allow for users to store and play songs as playlists, or play classified music at appropriate game points.

(23) The software shall have sound effects.

Game AI

(25) The software shall support builder AI.

(26) The software shall have AI replace a user when a user quits prematurely.

(29) The software shall support platformer AI.

Graphics

(27) The software shall support 2D graphics.

(28) The software shall show home screen (lobby) before and after games.

Performance Requirements

Design Constraints

Other Requirements

Appendix A: Use Cases

Use Case List

(1) Player starts the program

(2) Player connects to a session

(3) Player starts platformer game

(4) Player starts builder game

(5) Player plays platformer game

(6) Player plays builder game

(7) Builder wins

(8) Platformer wins

(9) Player quits

(10) Player breaks a rule

(11) Player changes settings

(12) Player creates a user account

Use Cases

---

(1) Player starts the program

- **Requirement(s) explored:** 1, 16, 24, 28

- **Player (actor) context (role):** All Roles
- **Preconditions:** The player has installed the program on their computer.
- **Trigger(s):** The player runs the program.
- **Main course of action:**
    1. Player sees a login screen.
    2. Player fills in their user information and presses the submit button.
    3. The game server authenticates the login information
    4. Player sees the game lobby (Use Case 2)
- **Alternate course(s) of action:**
    2. Player selects the option to create a user account (Use Case 12) Or:
    3. The game server rejects the login information
    4. The player is returned to the login screen.
- **Exceptional course(s) of action:**
    2. Player chooses not to play the game, and quits.

(2) Player Connects to a Session

- **Requirement(s) explored:** 0,1,4,7,16
- **Player (actor) context (role):** Player
- **Preconditions:** The player wants to create or join a multiplayer game
- **Trigger(s):** The player enters the multiplayer lobby
- **Main course of action:**
    1. The player chooses to create a new session
    2. The player selects a map for the game
    3. The player either invites specific players or opens the game to anyone in the lobby
    4. The player selects a role (platformer or builder) to play as.
    5. The player waits for other players to join.
    6. The player stops waiting, allowing AI characters to fill in any roles not filled by real players.
- **Alternate course(s) of action:**
    - Joining a game
        1. The player joins one of the open sessions in the lobby
        2. The player selects a role (platformer or builder) to play as.
        3. The player waits for others to join .
        4. The player confirms that they are ready when the creator of the session decides to stop waiting for players to join.
    - Alternatively: Instead of allowing the players to choose their role, the program tries to balance the game by choosing roles for them (Requirement 3). This would probably be an option for the session creator, but there's no way to know for certain since this isn't discussed in the design document.
- **Exceptional course(s) of action:**
    - The player abandons a game before enough others join
    - The player's connection is dropped
    - The player attempts to join a game that already has enough players.

(3) Player starts platformer game

- **Requirement(s) explored:** 7, 8, 9, 10, 11, 13, 18, 27

- **Player (actor) context (role):** Platformer
- **Trigger(s):** Player chooses to play a game acting in the platformer role.
- **Main course of action:**
  1. The player's screen displays the platformer view of the map, lives, and items.
  2. Background music for the level plays.
  3. After a certain period of time, the player moves along the level (Use Case 5)
- **Exceptional course(s) of action:**
  - Game crashes

(4) Player starts builder game

- **Requirement(s) explored:** 5, 6, 7, 9, 10, 11, 12, 18, 27
- **Player (actor) context (role):** Builder
- **Trigger(s):** Player chooses to play a game acting in the builder role.
- **Main course of action:**
  1. The player's screen displays the builder view of the map, tower options, and resource information.
  2. Background music for the level plays.
  3. The player is given the options to select a tower to place or zoom in on the map.
- **Exceptional course(s) of action:**
  - Game crashes

(5) Player plays platformer game

- **Requirement(s) explored:** 0, 8, 10, 13, 14, 16, 26, 29
- **Player (actor) context (role):** Platformer
- **Trigger(s):** Player has started a game as a platformer, and is beginning movement
- **Main course of action:**
  1. Player works his/her way towards the goal
  2. Player jumps, climbs, and runs to get around obstacles
  3. Background adjusts to show player a specific portion of the map
- **Alternate course of action:**
  1. Player picks up items
  2. Player uses item
  3. Player gains power-up (effective immediately)
  4. Player dies
     a. Player is respawned at most recent checkpoint with one less life
     b. Player has no more lives, and it out of the game (given builder view)
     c. Player has no more lives, but is donated a life by another player and is thus respawned at the last checkpoint
     d. Player has no more lives and steals a life from an AI player
- **Exceptional course(s) of action:**
  1. Game crashes
  2. Player quits, and is replaced by AI

(6) Player Plays Builder Game

- **Requirement(s) explored:** 1 5 6

- **Player (actor) context (role):** Builder
- **Trigger(s):** Player has started game as a builder
- **Main course of action:**
    1. Player attempts to place obstacles in the path of the Platformer(s) to impede their progress
    2. Player selects new object and places it on the map
        a. Player's amount of resources shrinks by the cost of item
    3. Player selects object on map and deletes it
        a. Player's amount of resources increases by the redemption value of item
- **Alternate course(s) of action:**
    1. Player selects new object but does not have enough resources to buy it. A warning message appears.
- **Exceptional course(s) of action:**
    1. Game crashes
    2. Player quits and is replaced by AI

(7) Builder Wins

- **Requirement(s) explored:** 16, 23, 27, 28
- **Player (actor) context (role):** Builder
- **Preconditions:** The game has been started and is still in progress.
- **Trigger(s):** Platformer player(s) do not reach the end of the level in the allotted time, or platformer player(s) die before completing the level.
- **Main course of action:**
    1. All screens "freeze" - users have no more control over their game pieces or player
    2. A message pops up on the screen, displaying the name of the winner, how long the game lasted, and any other relevant statistics, as well as two buttons - "okay" and "quit".
    3. User presses the "Okay" button, and is taken back to the main lobby screen (Case # ???)
- **Alternate course(s) of action:**
    3. Player presses the "quit" button, and the game closes (Case # ???)
- **Exceptional course(s) of action:**

(8) Platformer Wins

- **Requirement(s) explored:** 16, 23, 27, 28
- **Player (actor) context (role):** Platformer
- **Preconditions:** The game has been started and is still in progress.
- **Trigger(s):** Player reaches the end of the level with lives remaining
- **Main course of action:**
    1. All screens "freeze" - users have no more control over their game pieces or player
    2. A message pops up on the screen, displaying the name of the winner, how long the game lasted, and any other relevant statistics, as well as two buttons - "okay" and "quit".
    3. User presses the "Okay" button, and is taken back to the main lobby screen (Case #

???)

- **Alternate course(s) of action:**
  3. Player presses the "quit" button, and the game closes (Case # ???)
- **Exceptional course(s) of action:**

(9) Player quits

- **Requirement(s) explored:** 1, 16, 26
- **Player (actor) context (role):** All roles
- **Trigger(s):** Player clicks on 'exit game' button or game crashes.
- **Main course of action:**
  1. Software notifies other players in the game that player has quit.
  2. The server takes over from the point the player has quit with a replacement AI player (builder or platformer).
  3. User is taken to lobby page and has chance to start another game (if player quit and the game didn't crash)
- **Exceptional course(s) of action:**
  1. Software crashes while player is in the middle of playing the game.

(10) Player Breaks Rule

- **Requirement(s) explored:** 5, 17
- **Player (actor) context (role):** Builder
- **Preconditions:** Game is in progress.
- **Trigger(s):**
  1. The builder attempts to put an obstacle in a location that will make it impossible for the platformer to win.
  2. (alternative) Same as above, except this is the second time that the builder has tried to do this with the same obstacle.
- **Main course of action:**
  1. Obstacle is highlighted red and flashes twice.
  2. The player looks for a new place to put the obstacle.
- **Alternate course(s) of action:**
  1. Obstacle is highlighted red.
  2. An alert box appears with text along the lines of "You aren't allowed to place that obstacle here! (Explain...)". The "(Explain...)" text would be clickable.
  3. Upon clicking the (Explain...) text, the box would expand to show a mini-tutorial on obstacle placement. This would consist of text explaining that the builder is not allowed to completely block a platformer from completing a level, and a picture explaining the green/red highlights that appear on the map to help the builder know where they are allowed to place an obstacle once they select it.
- **Exceptional course(s) of action:**
  1. The game allows the builder to place the obstacle.

Appendix B: Requirements Matrix

Req #	Component(s)	Use case(s)	Priority (high,med,low)	Status (% complete)
0		2,5	high	100%
1		1,2,9	high	100%
2			low	0%
3			low	0%
4		2	high	20%
5		4,11	high	100%
6		4	med	0%
7		2,3,4	high	100%
8		3,5	high	90%
9		3,4	med/low	50%
10		3,4,5	med	100%
11		3,4	high	100%
12		4	high	100%
13		3, 5	high	100%
14		5	high	100%
15			low	0%
16		1,2,5,7,8,9	high	50%
17		11	med/low	0%
18		3,4	high	100%
19			low	0%
20			(1) high (2) low	50%
21				90%
22				50%
23		7,8	high	100%
24		1	low	0%
25			low	0%
26		5,9	low	0%
27		3,4,7,8	high	100%
28		1,7,8	low	0%
29		5	high	0%