Champion Document
Company: ScoSoft | Game: Scoto | Feature: Obstacles
Name: Kadence Kee | Date: 9/23/2021

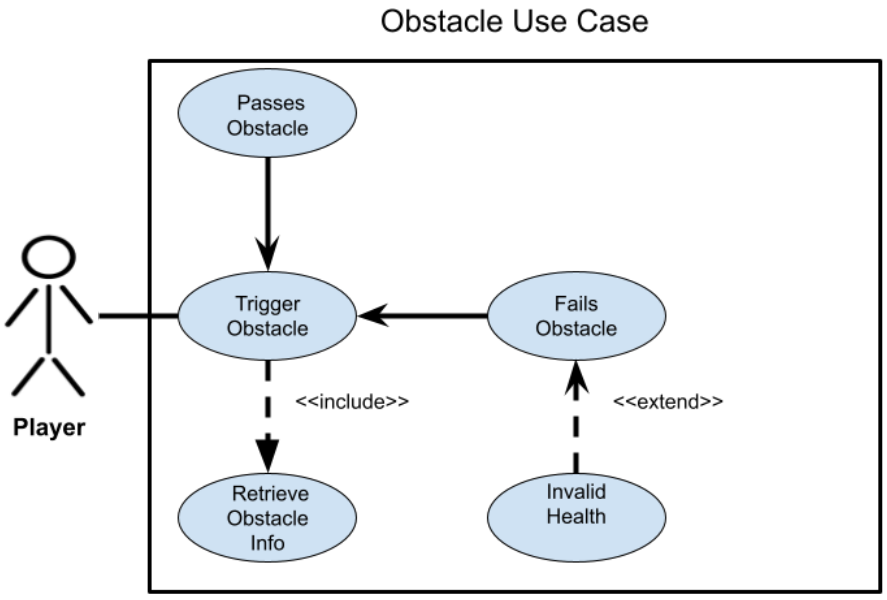
1. Brief introduction

My feature for the Scoto video game is obstacle design and implementation. The first version of the game will have three types of obstacles: a basic tripwire, rolling rock, and a trap door. The traps will be randomly spawned in certain rooms throughout the maze at the beginning of every level.

When the player object approaches the vicinity of the obstacle it will be triggered. The player will then have a certain response time to press the correct key to either dodge or jump to avoid the obstacle. If the timer runs out or the wrong key is pressed, the player will lose health. If health is not zero, the player can continue game play. If health is zero the player must restart the level. If the player is successful in pressing the right key then the player can continue. Later versions may include more complicated traps.

2. Use case diagram with scenario

Use Case Diagrams



Scenarios

Scenario 1 (Obstacle Use Case Diagram):

Name: Pass Obstacle
Summary: Player successfully enters the correct key in time and safely avoids the obstacle.
Actors: The Player of the game.
Preconditions: The obstacle was successfully loaded into the room and has not been triggered by the player.
Basic sequence:

- **Step 1:** Player object triggers obstacle.
- **Step 2:** Read obstacle type data from obstacle class.
- **Step 3:** Enable obstacle UI and begin timer.
- **Step 4:** Player responds by pressing the correct key before the timer runs out.
- **Step 5:** Signal that player passed the obstacle.
- **Step 6**: Player continues with game play.

Post conditions: Player health does not equal zero.
Priority: 2
ID: K01

Scenario 2 (Obstacle Use Case Diagram):

Name: Fail Obstacle
Summary: Player fails to enter the correct key in time and gets caught in the obstacle.
Actors: The Player of the game.
Preconditions: The obstacle was successfully loaded into the room and has not been triggered by the player.
Basic sequence:

- **Step 1:** Player object triggers obstacle.
- **Step 2:** Read obstacle type data from obstacle class.
- **Step 3:** Enable obstacle UI and begin timer.
- **Step 4:** Player does not respond and the timer runs out.
- **Step 4**: Player presses incorrect keys and timer runs out.
- **Step 5:** Signal that player failed the obstacle.
- **Step 6:** Subtract health from the player.
- **Step 7**: Player continues with game play.

Exceptions:

- **Step 7:** End game if health < 0

Post conditions: Player health equals zero.
Priority: 2
ID: K02

3. Data Flow diagram(s) from Level 0 to process description for your feature

Data Flow Diagrams

Diagram 0

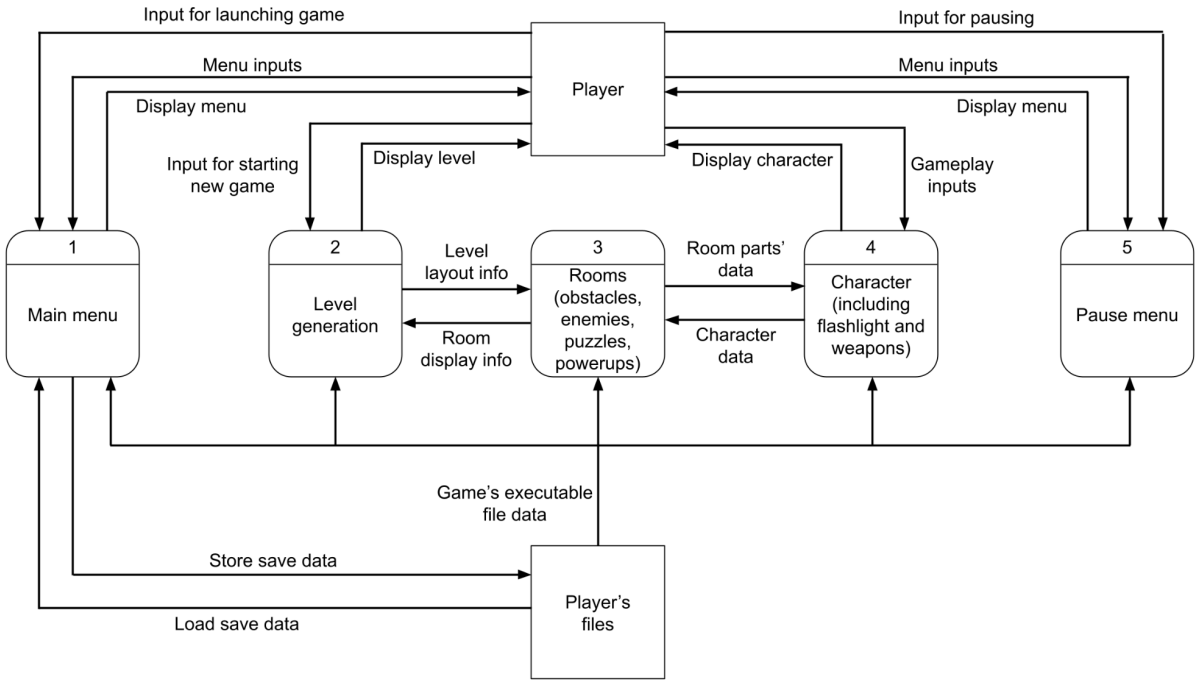
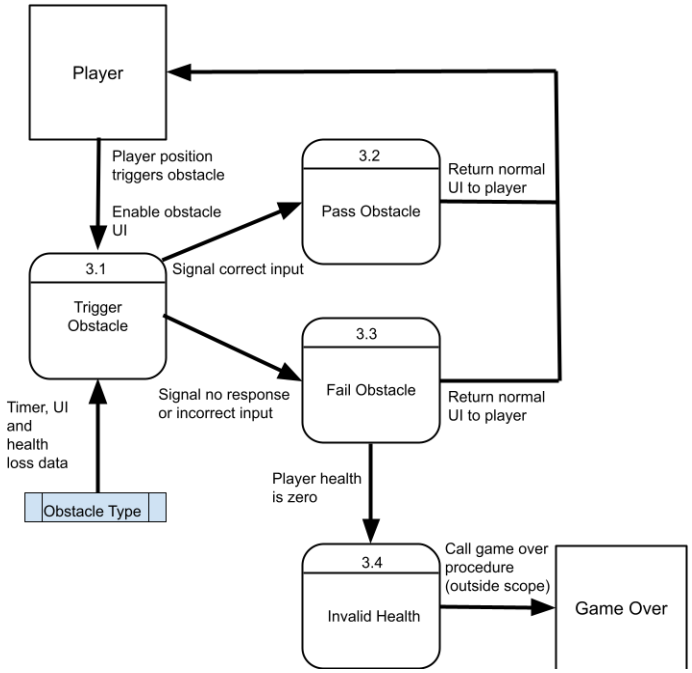


Diagram 3.1

Obstacle Subdiagram



Process Descriptions

Trigger Obstacle()
Retrieve obstacle type data from obstacle class database
Read user input
IF correct input
    Pass obstacle()
ELSE
    Fail obstacle()
Return normal UI to player

Fail obstacle()
    Lose player health
    IF player health < 0
        Game over

4. Acceptance Tests

Correctly load obstacle class information.

1. Test that normal UI is disabled and obstacle UI is called when obstacle is triggered.
2. Check that obstacle type data is successfully loaded from obstacle class database.

Fail obstacle.

3. Test that no input triggers fail obstacle .
4. Test that incorrect input triggers fail obstacle.
5. Test that the timer running out triggers fail obstacle.
6. Check if player health is successfully updated on fail obstacle.
7. Ensure that if health < 0 that the game over procedure is called (boundary case)
8. Ensure that if health > 0 normal UI is restored.

Pass obstacle.

9. Test that the correct input triggers pass obstacle.
10. Ensure pass obstacle restores normal UI to player.

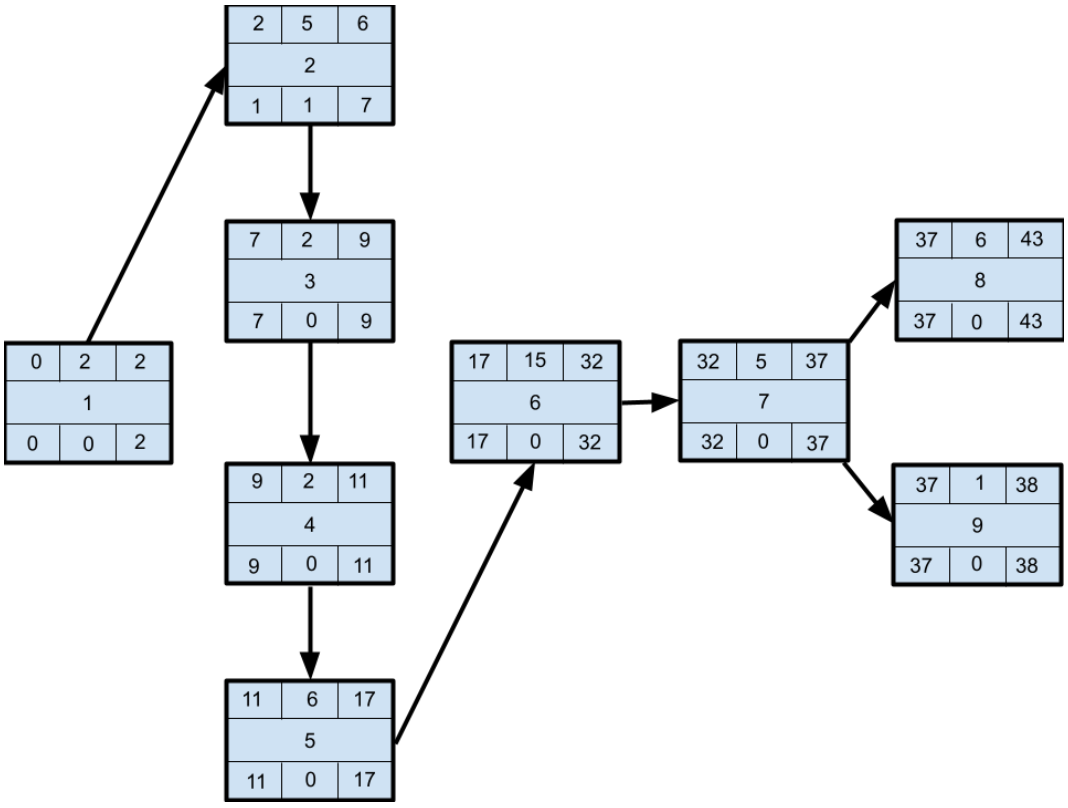
5. Timeline

Work items

Task	Duration	Predecessor
1. Requirements Definition	2	-
2. Obstacle Design	5	1
3. Use Case Design	2	2
4. DFD Design	2	3
5. UML Design	6	4
6. Obstacle Programming	15	5

7. Testing	5	6
8. Documentation	6	7
9. Installation	1	7

Pert diagram



Gantt timeline