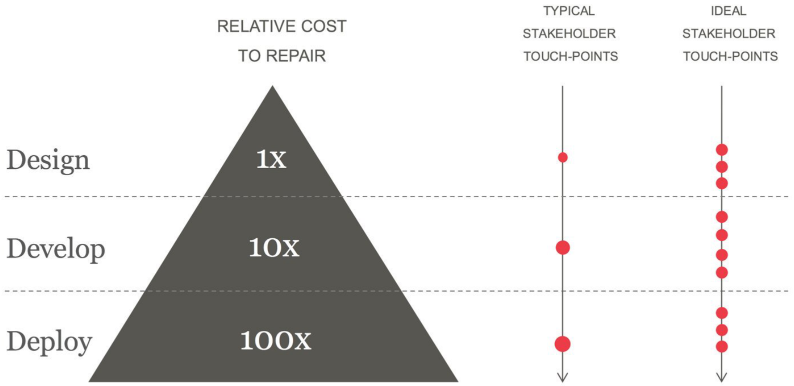
Design

Why Design?


RELATIVE COST TO REPAIR
TYPICAL STAKEHOLDER TOUCH-POINTS
IDEAL STAKEHOLDER TOUCH-POINTS
Design
1X
Develop
10X
Deploy
100X

Much easier to fix issues during the design phase.

Software Design

- What should the software do?
- What are the components of the program?
- How will you represent those components?
- How will the different components interact with one another?

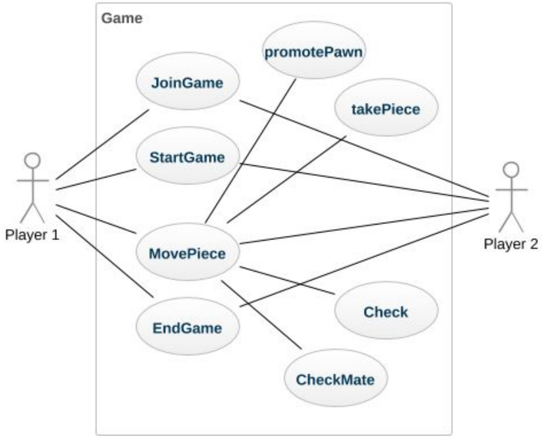
How do you communicate the software design clearly?

Unified Modeling Language

Somewhat standard tool for communicating software design.

Use Case Diagram

Describes the different types of users called **actors** and the **actions** that the users can take in a software system.

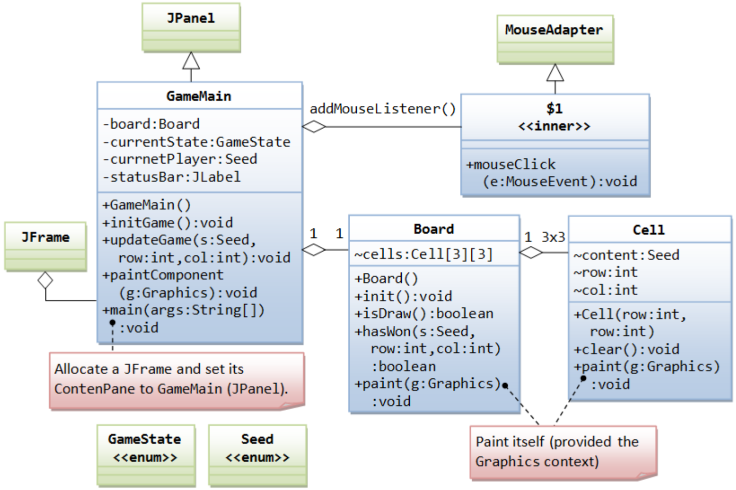


Use Case Diagram Actions

- Actions are approximately the same level of granularity.
- Each action begins with a verb (e.g. “make move” as opposed to “piece movement”)
- Each action is specific (e.g. “make move” as opposed to “change board”)
- Actions are written concisely.

Class Diagram

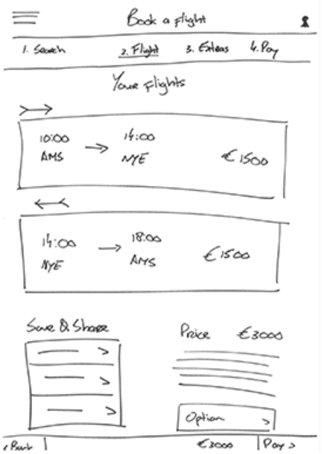
Describe the classes in the software system.

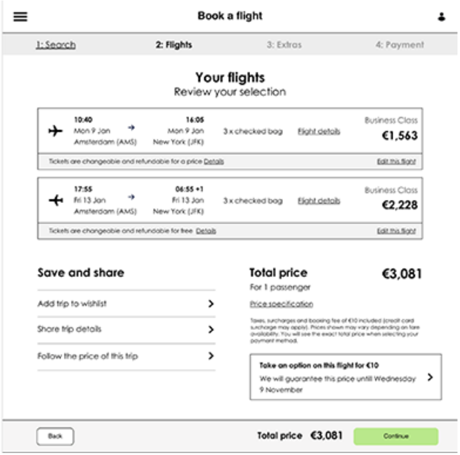


What are the different components? How do they interact with each other?

Visual/Interaction Design: Wireframe

Visual guide that represents the skeletal framework of software. Describe how to arrange the functional elements, and how users can interact with them.





Design Review

Teams often review the design of the software, before writing a single line of code.

Goal: Sanity check and improve the design of the software.

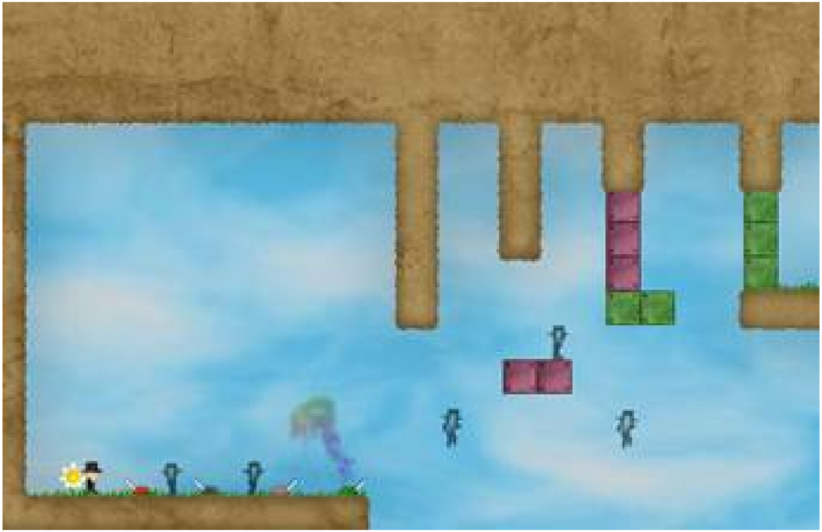
Your Group Project

Your second deliverable is a **design presentation**.

Normally, design reviews have a lot more audience interaction than we can afford.

Software Design Example

The Company of Myself



- http://mypuzzle.org/company-of-myself
- https://www.youtube.com/watch?v=tI0RfSn8oYg

How to Design This Game?

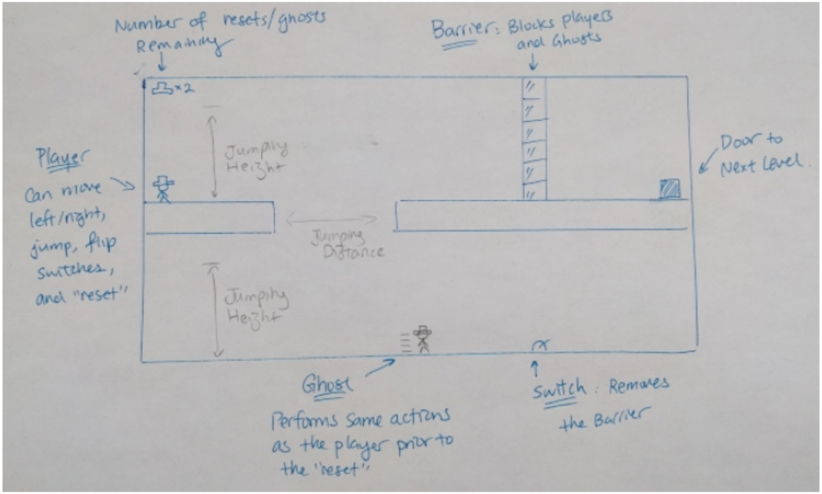
- Functional Requirements:
  - What can the player(s) do?
  - What are the game mechanics?
- Non-Functional Requirements:
  - Will the game lag?
- Visual/Interface Design:
  - How will the game look?
- Software Design:
  - What classes will we need?
  - What is the inheritance structure?

Player Actions

- Move left/right
- Jump
- (Jump at the same time as moving left/right)
- Flip a switch
- “Restart”
- “Make ghost”

Ghosts can do the same things except “Restart” and “Make ghost”.

Wireframe


Number of resets/ghosts Remaining
×2
Barrier: Blocks players and Ghosts
Player
Can move left/right, jump, flip switches, and "reset"
Jumping Height
Jumping Distance
Jumping Height
Door to Next Level.
Ghost
Performs same actions as the player prior to the "reset"
Switch: Removes the Barrier

Classes

- Something to control the board
    - Finish level
    - Restart
    - Check for barriers
    - The main “tick” function

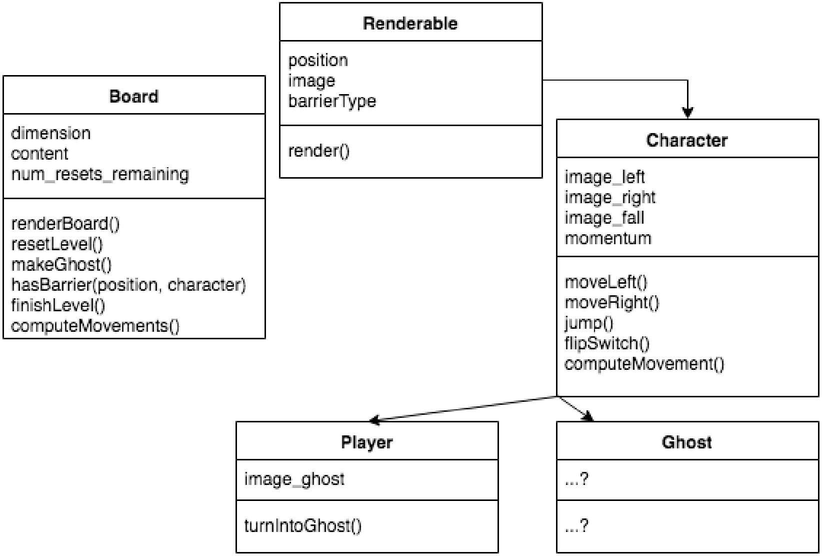
Classes

- Something to control the board
    - Finish level
    - Restart
    - Check for barriers
    - The main “tick” function
- A superclass for objects on the board (doors, switches, etc)

Classes

- Something to control the board
  - Finish level
  - Restart
  - Check for barriers
  - The main “tick” function
- A superclass for objects on the board (doors, switches, etc)
- Character like players / ghosts that can move

Class Diagram (WIP)



Not perfect, but is a communication tool to help design software.

Class Diagram

There are many decisions to be made about how to structure the classes!

A class diagram provides an overall picture of the software design.

A class diagram does **not** answer all important design questions.

Data Representation

- How will you represent the board?
- How will you represent the walls and floors?
  - What about walls and floors with complex shapes?
- At what point will a character run into the wall?
- What order will you render things?

Computing Movement

At every “tick”, we compute the character movements:

- Are there any key presses?
- Is the level complete?
- Do the ghosts move?
- Does the player move?

Other questions we have to answer:

- How do we store the player movement?
- Can characters move at the same time as flipping a switch?

Ordering

- Check if the level is complete (player or a ghost is at the door)
- Check for any key presses
- Check for “reset” (make ghost)
  - Create a new ghost
  - Reset tick counter and ghosts to an initial state
- Check if user wants to restart the level
- Check if a character is flipping a switch
- Move each character (what order?)
  - Is there momentum from a previous jump?
  - What if a character hits the ceiling?
  - Is there a new jump? Can a player jump?
  - Move left or right?
- Save the key press

Saving Player Movement

Should the **Player** class track movement, or the **Board** class?

Do we save the **key presses** or the **pixel-wise movement** at each tick?

These are hard design decisions that can have ramifications on what kind of levels we can design.

Communication and Critical Thinking

- Making these design decisions is often an open-ended problem requiring critical thinking.
- Determining the best decision requires clearly explaining the advantages and disadvantages that you notice.
- Getting everyone on the same page can be difficult, and requires interpersonal communication skills.
- Catching issues during the design phase is much better than catching them later on!

Splitting the Work

What are the major pieces?

- Board
    - Rendering
    - Check if there is a barrier
    - Make ghost
    - Restart
- Character
    - Moving left and right
    - Jumping (momentum)
    - Falling (with momentum?)
- Ghost
    - Creating a Ghost from movements
- Player
    - Recording movement

Ordering the Tasks

Try to make the game playable as quickly as possible.

For example, implementing switches and removable barriers can come after movement.

Breakdown and Design

- The better your group communicate the design, the easier it is to work on different portions in parallel.
- If you are waiting for each other a lot, then you have not communicated the design well enough.

Fixing The Design

What to do if you make the wrong design decision?

- Try to catch and fix issues early.
- But, changes can affect multiple parts of the code.
- Keep all group members in the loop.
- Decide what process is to change the design while keeping everyone in the loop.