

Problem Decomposition Revisited (Again and Again...): Object Oriented Design

There's more...?



Outline

- **Object Oriented Design**
 - Identify the Classes
 - Identify what Information each Class Needs
 - Identify what each Class Needs to Do



Software Development Life Cycle

1. Understand the Problem = Requirements Analysis
2. Work out the Logic = Design
3. Convert it to Code = Implementation
4. Test/Debug
5. Maintenance

Today we will start to analyze and design the solution to a program.



What are the Nouns?

- **The Game.** The Wumpus World game takes place in a cave with different rooms in it. You can think of the cave as an NxM rectangular grid. The player always starts in position 0,0, which is guaranteed to be safe (but it may still be smelly or breezy or glittery).

The objective of the game is to find the gold. The player will know when he/she is in a room with the gold because there will be a "glitter" in that room. If the player detects a glitter, he/she can pick up the gold and the game is won.

Bottomless pits are present in some of the rooms. There is a 20% chance that any given room will have a pit. All rooms adjacent to a pit are breezy, that is, a player entering a room adjacent to a pit will detect a breeze. If the player moves into the room with a pit, he/she falls in and dies a horrible death.

There is only one wumpus in the cave, and he is also placed at random. Rooms adjacent to the wumpus are smelly, that is, a player will detect a stench in a room adjacent to a wumpus. The wumpus cannot move. If the player enters a room with the wumpus, he/she will be eaten, and, once again, die a horrible death.

There is also only one room in the cave that contains the gold. Unlike the other objects, the player has to be in the same room as the gold in order to detect a glitter. Like the wumpus, the gold is placed at random.

The player can move up, down, left, or right. The player also has one arrow. Once it's used up, it's gone. It can be used to shoot a wumpus, and can be shot in any direction the player can move in. If the player is successful in shooting the wumpus, the wumpus will emit a blood-curdling scream, and will no longer be a threat. The only other action the player can perform is to "grab gold".

When the player first starts the game, he/she does not know (and cannot see) where the location of pits, gold and the wumpus are. The only clues are whether the current room is breezy, smelly, or glittery.

Test Cases I came up with -- Class Suggestions on Next Slides

- **Does the cave draw as NxM?**
- **Does player always start in 0,0?**
- **Does player win when he/she picks up gold?**
- **Do pits appear in 20% of the rooms (on average)?**
- **Are rooms adjacent to pits always breezy?**
- **Does player always die if he/she falls into a pit?**
- **Is the wumpus always placed randomly?**
- **Are rooms adjacent to the wumpus always smelly?**
- **Does player always die if he/she enters a room with a wumpus?**
- **Is the gold always placed randomly?**
- **Does the player always move in the correct direction?**
- **Will the program catch it if the player attempts to move outside of the cave?**
- **When the player shoots an arrow, is it removed from “inventory”?**
- **If the player shoots in the direction of a wumpus, and the wumpus is in a direct line, does it always die?**
- **If a wumpus dies, does the stench go away?**
- **Are unvisited tiles drawn as unknown?**
- **Are visited tiles drawn as “lit”, or known?**

1.) If wumpus is in 0,0 - FAIL

2.) If pit is in 0,0 - FAIL

3.) Uses correct dimensions

4.) Player starts in 0,0 always

5.) If glitter then gold

6.) If gold picked up, Player wins

7.) # of pits should be 20% on average

8.) All adjacent rooms must be breezy

9.) Player dies if he/she falls in pit

10.) On

11.) Adj

12.) If

13.) If

14.) On

15.) I

d

16.) One

17.) No

18.) Kills

10) Only one wumpus

11) Adjacent rooms must be smelly

12) If wumpus dies, stench goes away

13) If player encounters wumpus he/she die

14) Only one gold

15) If keys pressed, player moves in that direction, If it's in the boundaries of cave

16) One arrow to start

17) No arrows after arrow is shot

18) Kills wumpus if shot in that direction

19) Arr

20) Une

21) If
St

- 19) Arrow doesn't go out of bounds
- 20) Unexplored spaces are not shown
- 21) If player has been in room it is shown

What are the Nouns?

The Game. The Wumpus World **game** takes place in a **cave** with different **rooms** in it. You can think of the **cave** as an NxM rectangular **grid**. The **player** always starts in **position** 0,0, which is guaranteed to be safe (but it may still be smelly or breezy or glittery).

The objective of the **game** is to find the **gold**. The **player** will know when he/she is in a **room** with the **gold** because there will be a "glitter" in that **room**. If the **player** detects a glitter, he/she can pick up the **gold** and the **game** is won.

Bottomless **pits** are present in some of the **rooms**. There is a 20% chance that any given **room** will have a **pit**. All **rooms** adjacent to a pit are breezy, that is, a **player** entering a **room** adjacent to a **pit** will detect a breeze. If the **player** moves into the **room** with a **pit**, he/she falls in and dies a horrible **death**.

There is only one **wumpus** in the **cave**, and he is also placed at random. **Rooms** adjacent to the **wumpus** are smelly, that is, a **player** will detect a stench in a **room** adjacent to a **wumpus**. The **wumpus** cannot move. If the **player** enters a **room** with the **wumpus**, he/she will be eaten, and, once again, die a horrible **death**.

There is also only one **room** in the **cave** that contains the **gold**. Unlike the other objects, the **player** has to be in the same **room** as the **gold** in order to detect a glitter. Like the **wumpus**, the **gold** is placed at random.

The **player** can move up, down, left, or right. The **player** also has one **arrow**. Once it's used up, it's gone. It can be used to shoot a **wumpus**, and can be shot in any direction the **player** can move in. If the **player** is successful in shooting the **wumpus**, the **wumpus** will emit a blood-curdling **scream**, and will no longer be a threat. The only other action the **player** can perform is to "grab **gold**".

When the **player** first starts the **game**, he/she does not know (and cannot see) where the location of **pits**, **gold** and the **wumpus** are. The only **clues** are whether the current **room** is breezy, smelly, or glittery.

Cave
rooms
width
height

20% of rooms
are pits

Wumpus location

Gold location

draw

initialize itself ←

toString

Player

x, y position

arrow

move

draw

init

shoot

grabGold

getX

getY

getArrow

Room

visited or not
breeze
stench
gold
wumpus
pit

draw
toString
setConditions
getConditions
init

Class Cave

draw(self)
--init-- (self, width, height)
string toString(self)

Class API's

Class:	Returns	Method	Parameters	Description
Room		<code>__init__</code>		Construct a room
		<code>setCondition</code>	condition, value	Set the condition of a room (e.g. breezy, has a Wumpus, etc.)
	boolean value	<code>getCondition</code>	condition	Returns the value of a room condition
		<code>draw</code>	image size, x, y	Draws the room image with contents at the size and x, y location
Cave		<code>__init__</code>	width, height	Constructs an NxM cave and adds all objects (pits, breezes, etc.)
		<code>draw</code>		Draws the cave with all its rooms
Player		<code>__init__</code>		Constructs a player
	int x	<code>getX</code>		Returns the x location of the player
	int y	<code>getY</code>		Returns the y location of the player
	boolean	<code>getArrow</code>		Returns whether the player has an arrow left or not
	boolean	<code>move</code>	height, width, direction	Changes players location and returns True if successful, False otherwise
		<code>shootArrow</code>		Removes the arrow from inventory
		<code>draw</code>		Draws the player at his/her current location

More Tests?

- Write tests that will exercise the defined methods in the APIs

Summary

- **Object Oriented Design**

- Identify the classes
- Identify what information each class needs
- Identify what each class needs to do
- Identify use cases
- Define the API
- Define the instance variables
- Finally – write some code!

