In your code for the Gauntlet game of HW1, explain how you initialized the rival players and restricted their position to either side of the screen.

I made another constructor for the rivals that took a second parameter int that represented how many rivals were on the screen. I then took mod 2 of the int to determine if the rival should spawn on the left or right. If it's supposed to spawn on the left I subtracted \( \frac{2}{3} \) the window's x size from the hi point along the x-axis when calling setMoveBox. If the rival is supposed to spawn on the right I added \( \frac{2}{3} \) window's x size to the lo point along the x-axis.
Resolution to practice midterm

Problem #2

Player Movement:
For example, Pac Man is restricted to 1.25 dimensions of movement because he is usually restricted to one axis of movement except at intersections where he’s allowed to change to the other axis of movement.

Dimensionality of the world:
The depth of the terrain of the game world and its inhabitants, such as 2D or 3D sprites and walls, dictates the amount of dimensionality of the world.
For example, the original Mario Bros. game allows walking left/right & vertical jumping, but Mario Sunshine allows movement in a fully 3D world where you can walk in all four cardinal directions & jump up.

Game’s viewpoints dimensionality:
The angle, position, and “zooming” capability of your “camera” on the game world influences the dimensionality of your viewpoint. Doom allows a fully changeable 3D viewpoint while games like the original Metroid only have 2D scrolling, or even have no change of viewpoint in games like Pong.
3) A use case diagram is a visual tool to help better understand the needs of the end user. Actors are represented by stick figures. The program is represented as a big rectangle. Ovals are use cases.

```
Player

watch
play
load

Programmer

extend
test
```

10) draw → feelister

animate

collide

move

update

Note if look at source code much more detailed things happen, but this was my intended answer.
#5 Noun-Verb Analysis

Map nouns in the requirement description to class names.
Map verbs in the requirement description to member functions.

ex.
the player must move back & forth to avoid bullets shot by bad guys. The player may eat food to regain health

<table>
<thead>
<tr>
<th>Player</th>
<th>Bad Guy</th>
</tr>
</thead>
<tbody>
<tr>
<td>eat()</td>
<td>shoot()</td>
</tr>
<tr>
<td>move()</td>
<td></td>
</tr>
</tbody>
</table>
6. Draw the UML diagrams of:

- Command, Singleton, and Bridge

and explain how they are used in Pop.

- **Command**
  - Command Pattern
  - `doCommand(Command cmd)`

- **Bridge**
  - Context
    - `Interface = Bridge`
  - Interface
    - `virtual alg1(...)`
    - `virtual alg2(...)`
    - `...`
  - Implementation 1
    - `alg1(...)`
    - `alg2(...)`
    - `...`
  - Implementation 2
    - `alg1(...)`
    - `alg2(...)`
    - `...`

- Pop uses the `ServiceRequest struct`, which passes off `command execution` to `乙方`.

- Pop has a `ClassLoader` to bridge to either MFC or Open GL graphics.
Singleton

```java
private static Singleton _instancesSingleton;

private Singleton();

public static Singleton * pInstance;

if (_instancesSingleton == NULL)
    _instancesSingleton = new Singleton();

return _instancesSingleton;
```

Pop uses the Singleton pattern in the Randomizer class.
8) update():
  calculates all forces on critters in an array.
  move(dt)
  loops through the array and changes the critters' position based on the total forces on it.

It appears to move in parallel because all these calculations and movements are done before it is drawn to the screen and when it is drawn all the movements have already occurred.

Since update calculates forces w/o moving the creatures and presumably the new forces only depend on current values of positions, velocities, order of calculating critters in update does not matter.

Since move calculates new positions/velocities based only on the already updated forces, order in this case doesn't matter.

The fact that both these operations are order independent gives rise to the feeling of parallelism in the game, because we can't see how observing the game what order things were updated internally.
Problem 9: Draw an $x$ by $y$ playing field on a piece of paper like the

Randomly choose $x$ points on the boundary (not all points), so your chosen points

Randomly create a rectangle, if none are removed if it is too small.

Randomly create about 10 rectangles.

Draw with a system around the room to accommodate balls, any edges from

rectangle to connect to the edge of the other room (gray, etc.).

This is how you connect the rooms.

After that, connect the doors.