Cevat Aykan Sevinc aykan.sevinc@ug.bilkent.edu.tr

Question 1)

In your opinion, what are the different development phases we should go through and what percentage of total development time should each phase take?

->

I believe there are five main development phases in an iteration. Firstly, there is the Analysis phase. We must understand the game we are going to develop. The functional and non functional requirements in the game with the idea of the game must be examined. What are the rules in the game, what is the goal of the game, are there any characters, what actions can players do and how do players interact with the game are some questions to answer during analysis.

Secondly, there is the design part. In this phase, we must design levels, art and gameplay. For levels and art, we can design prototypes that are a representation of the concept level design and art. For the gameplay, we must understand the objects in the game, how do they interact with each other and apply design patterns for a maintainable and extendible codebase. We also need to consider non-functional requirements during the object design. There may be drawbacks when favoring a non-functional requirement. Such a drawback can be the usability if the game has too much features.

Thirdly, there is the implementation. we must implement and combine the art design, level design and the object design in the game to realize the game.

Next, there is the testing phase. In this phase, we must make sure the game is covered and it passes functional and non-functional requirements. This phase can be combined with the implementation by test-driven development.

Lastly, there is the acceptance phase. If there is not any error detected error in the software, the game can be released. If there are problems, we can iterate starting from the analysis.

Each phase is very important. But I think the analysis and the design are the most important phases. This is because if you do not understand the game you are going to develop, then the end result will not be the imagined game. If the design of the game is bad, then the maintainability and extensibility of the game can cause many problems in terms of change. The game may not cope with the change in future releases. Therefore, I would suggest to distribute development time by:

- Analysis: 15%
- Design: 20%
- Implementation: 40%
- Testing: 20%
- Acceptence: 5%

Cevat Aykan Sevinc aykan.sevinc@ug.bilkent.edu.tr

Question 2)

We would like to get a glimpse of the gameplay of our new game at the earliest.

- What are the first 5 mechanics/functionalities you should develop for this specific game so that we have a playable build as soon as possible?
- For each of the mechanics/functionalities you have listed in the previous section, what questions should the game designer have answered before you start coding?

->

We need to implement the functionalities required in the game for the player to play. These can be:

1. Basic functionalities of the player: movement, interaction with the world.
2. Basic functionalities of the world: how the world responds to the player.
3. Demo level: Prototype of a demo level that player can perform each functionality to test gameplay.
4. Basic character prefab: to represent our character in the game.
5. Basic world prefabs: world objects such as items, enemies or player interact-able objects.

->

1. We need to understand both the high and low level design of the character. We must understand what the player can do, what states it can be in, and design a low level implementation to realize how it can perform its functionalities with respect to other game objects.
2. Same with the player, we must understand the high and low level design of the world and create an object hierarchy to realize how each object interacts with each other.
3. We need to understand the environment in the game and design a level accordingly.
4. We need a basic prefab for the character to control for testing the implementation.
5. We need a basic world object prefabs to place in the demo level for testing interaction.

Question 4)

- Describe your strategy to decouple UI/Input modules from the rest of the game so that platform specific code (input event detection, UI control etc) does not creep into gameplay modules (movement, jump, attack etc.).
  Hint: Try to isolate input/UI code from character prefabs.

->
We can have an Input manager which gets the input from the player and controls the main character accordingly. This way the input logic is separated from the main character logic. When we have different platforms, we can switch between platforms using the platform specific if statements in the input manager. This way, the change in the build environment will not affect the game logic. Using a publisher-subscriber pattern, we

can update the UI to display any information when specific events happen. For the input UI, the related code can be in the Input manager.

However, this may cause some maintainability problems if the player has too much states. We might want to add new features for the player in the future. This may create complex boolean logic in the Input manager (as can be seen in my code). One way to solve this problem is to use a finite state machine pattern in the main character design. Inside each concrete state, we can get the input and update the player accordingly. In a way, this is a representation of Input manager inside the main character class without interfering the main character logic.