For this project, we identified three stages that we think work well for presenting different levels of deliverables. For our basic commitment, we commit to having a playable game with 4 identifiable parts:

- Voice Control
    - This module will interface with a BRAM buffer that stores histogram values made by a given FFT module. Specifically, it will use the available histogram to determine the general pitch of the incoming signal.
- Obstacle Position
    - Will determine what obstacles should be visible and what obstacles are about to collide with the player.
    - In the basic commitment, obstacles will be generated by iterating through an array of obstacle heights that will represent fixed positions within the games coordinate system. In order to detect a collision, we will check for overlaps between player x value and the x value of the closest obstacle. If there is an x overlap, we will then check that the player is within the allowed vertical bounds.
- Game FSM
    - Will keep track of whether or not the game is in progress, the player's vertical and horizontal position, and the direction of the player's movement for the next frame.
- Graphics
    - Will contain many smaller modules like characterSpriteGenerator and obstacleSpriteGenerator that will together determine what the player sees on screen.

For our goal, we hope to make a more interesting/complete game with:

- Scoring
    - A counter module will keep track of how long the player stays alive and output the "score" to the hex display on the nexus
- Player controlled parameter inputs
    - speed adjustment, difficulty adjustment, volume adjustment, sensitivity adjustment
    - A module similar to the one in lab 4 will keep track of which value the user wants to change and stores the new value
- Replay mode will record a players position over the course of a game.
    - To do this, we add states to the non-game in-progress states differentiate between the first game and those where recorded games exist.
    - To actually record the movement, we will just keep track of whether the player moved up, moved down or stayed in the same place. This will only take 2 bits to store each frame refresh, but may still require us to find alternatives to the readily available memory.
    - Finally to replay, we will add a flag that tells the FSM to read the next move from memory instead of taking it from the voice control module.

For our stretch goals, we hope to add additional functionalities and game modes:

- Procedurally generated obstacles
    - Figure out values to introduce “randomness” that we can use as a seed and from there look into random generation of values. To add this to replay mode will involve recording the seed value along with the player movements.
- Audio during replay mode
    - Given that we know which direction the bird moved, we can play a tone back at the user which mimics the pitch observed during the game.
- Second axis of movement
    - Use the arrow keys to add more complexity to player experience! This will require restructuring inputs to many of our modules as they currently rely on a deliberate increase in the players x location.
- Vertically moving enemies
    - Currently, once an obstacle appears on screen, the player knows exactly where they must be in order to safely avoid collision. An enemy that moves up and down will add variety to the game play.