Oneiroi

Interim Report

Current Progress

We are somewhat in line with our progress schedule and are currently working on our desired target. In some points we got even further and implemented things from the high target (clouds, parabolic throwing, angled camera), but it is fair to say that we haven't completed our desired target yet because we haven't implemented building walls.
On the other hand, it might be a good idea to invest some more time into playtesting before we move on, to make sure the game is really fun and balanced.

Map and Terrain

Each map has an own subfolder in the "Maps"-directory inside which all of its content files are stored. A heightmap (stored as heightmap.png file) is used in order to add more detail to the terrain: The mesh of the map has a grid structure and the height level of each grid vertex is obtained from the color value of the corresponding pixel of the heightmap image (the darker the pixel's color, the higher the z-value of the grid vertex). The height value at arbitrary map points is then obtained by interpolation from the height values of the four grid vertices that surround the given point. The mapareas.png file encodes the different areas of the map: Walkable parts of the map are white whereas black parts correspond to unreachable areas. In order to have a visual transition between the walkable and the unreachable areas of the map, cliffs are added while loading the map. The cliffs start at the boundary between the walkable and unreachable area and are implemented by sampling the negative displacement of the boundary vertices along the z-axis from a parabola. In order to have a better collision detection between the player figures and the cliffs, it is necessary to obtain the normal of the map boundary at the boundary point that is closest to the player figure's position. The computation of this normal is implemented by first projecting the position of the player's figure onto the map boundary and then evaluating the normal at the projection point by interpolating the normals of the two adjacent map boundary vertices.

Things that were easier than expected

1. Importing .fbx into XNA worked flawlessly.
2. Adding music was very simple.
3. The Xbox Garbage Collector and performance are less of an issue than we thought after the lectures.

Things that were harder than expected

1. Dream neutralization is O(n^2), which can be slow. In our Xbox tests, we get away without any supporting data structure if we don't have more than 150 dreams, which is a good thing since using e.g. a grid would clutter the code. We have offloaded the work of dream-related computations into its own thread, but that means that we have to be careful with correctness.
2. Collisions:
    1. It was difficult to work with the bounding boxes and sphere as we had a single mesh rather than separate parts, which made bounding sphere too large while bounding boxes aren't so great due to their restriction with axis alignment.
    2. With arbitrary terrain shapes are somewhat tricky because it is hard to get something which is both *correct* and *not overly annoying* (don't get stuck, if the player tries to move somewhere illegal just correct the movement instead of refusing to move at all etc.)
    3. We reduced the problem to 2D and still can't guarantee that players are always in 100% legal positions in each frame, but the result is somewhat close enough.
3. Our level editor uses JSON. Importing JSON in C# is usually easy, since there is built-in functionality. However, the specific class for this task is not available in the .net compact framework.
   There are numerous JSON libraries for C#, but as we had to compile our project for the Xbox, we needed one which is open source, compatible with Visual Studio 2010 and compatible with the .net 4.0 compact framework.
   Our solution was to download an old version of JSON.net, tweak the compile flags for Silverlight and remove some more functionality until it finally compiled for the Xbox. As a result, hours of work went into importing something as simple as JSON on the Xbox.
4. Converting HTML "top" and "left" attributes from the level editor into our own coordinate system, which has 0 in the center, a flipped Y-axis compared to HTML and some more differences caused a bit of a headache.
5. Some weirdness on the Xbox and in XNA:
    1. File names and paths cannot be longer than 40 characters
    2. Garbage is caused by the weirdest things, e.g. moving the mouse or changing the music volume. It is hard to track garbage creation in code because writing to the console can also cause garbage (Heisenberg effect).

Random things that would be a good idea to polish

1. Animations: The walk cycles could be more realistic.
2. Humans: Awake/sleeping humans could have some kind of yellow/purple glow, in order to distinguish more quickly between awake and sleeping humans.
3. Shadows: There are no shadows as of now.
4. Obstacles: We could use a nicer model for the obstacles. There are great free models for greek pillars available.
5. Throwing: Is not really useful or fun for now, it's more fun to just rally more dreams and let all of them clash with the enemy's dreams by running into them. We will have to make it easier to throw and maybe add an indicator where the dream is going to hit the ground.
6. Add a nicer background for the game. Just plain dark blue isn't that attractive.

Detailed Progress Check

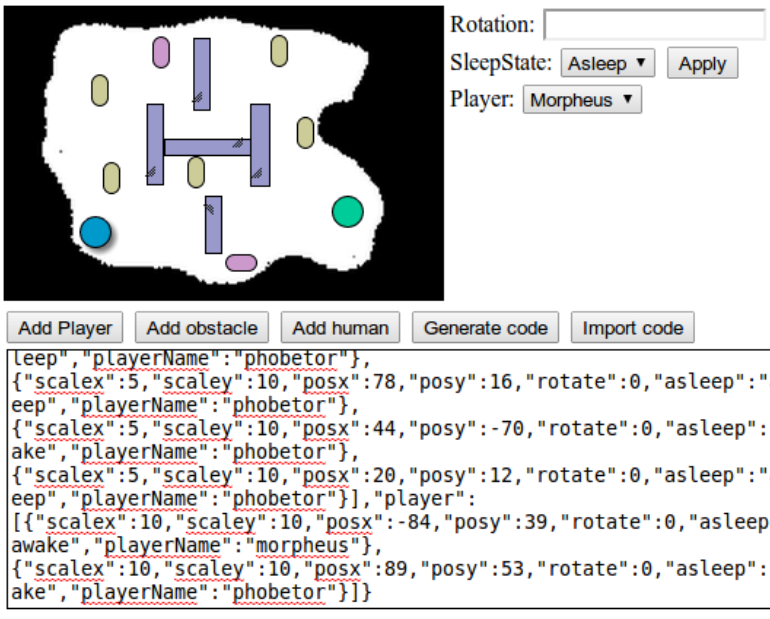
1. *Functional Minimum* - Complete
2. *Low Target* - Complete
3. *Desired Target* - In Process
    1. Effects of carrying Dreams - Remaining
    2. Dreams Animation and Shader - Complete
    3. Walls - Remaining
    4. Map textures - Complete
    5. Background music - Complete
    6. Sound Effects - Remaining
    7. Game Menu and Help Screen - Complete
4. *High Target* - In Process
    1. Obstacles (Random) - Remaining. We might not do it but have multiple maps instead.
    2. Throwing (parabola) - Complete
    3. Angled camera - Complete
    4. Clouds - Complete

Screenshots

Main menu screen

In-Game view

Debug mode: Collision boundaries, capture radii, load



Our simple JavaScript+HTML level editor