Dragonfly Engine Game Design Document

Brandon Contardi
bocontardi@wpi.edu

Ketan Chatterjee
klchatterjee@wpi.edu

Game Name: KleptoCosmonaut

Genre: Puzzle Stealth Heist Game

Premise: You are the famous **Dexter Driftstar**, thief extraordinaire who travels the universe stealing from evil doers. Today is no different, as **Vortex Corp** has created a means of controlling the entire universe with the **Singularity Seed**, a device that is devastating in the hands of such an evil and malicious company as Vortex Corp. The **Coalition of Planets** has enlisted your skills to board the Vortex Corp Star-Carrier ship and steal the Singularity Seed to stop the utter destruction of space civilization. You must traverse the different rooms aboard the star-carrier, avoiding detection from Vortex Corp guards, cameras, and avoid tripping the security laser alarms, etc. You must save the day once again, and bring safety to the universe.

Projected Tech Features: Predefined walk paths/cycles that game objects can traverse. Guards will use vision cones that detect the player and initiate game failure. Cameras that can swap which direction they are looking (while remaining stationary). Lasers that turn off and on that need to be passed. Levels will be prebuilt. Game is top down view.

Stretch Tech Features: Powerups throughout the map (freeze time, speed up character, etc).

Implemented Tech Features: 2 different types of guards, ones that stand still in a set location and swap the direction they are facing, changing the direction of their vision cones, another that patrols a set path and looks forward the whole path. On being spotted, the game initiates failure. Lasers that will fail the game if a player steps through them. Doors that can only be opened if the player first obtains a keycard.

Projected Artistic Features: Sprites for the hero, guard(s), and obstacles for the player to hide behind. UI elements for displaying a timer, current level, etc. SFX.

Artistic Stretch Features: Sprites for stationary objects such as cameras and lasers. If there is not enough time to get to these they will be indicated with solid colored shapes. Music.

Implemented Artistic Features: Sprites for the hero, guards, keycard, and the exits are all present.

Dragonfly Engine Game Design Document

Implementation Plan:

- Walk cycles will be their own class. The basis of the functionality will be a Vector array that will act as a list of points on the screen for the game object to traverse. Other than the normal set/get functions, there will be an addVector() function to more easily define the path. One limitation is that turns will probably have to be 90 degrees
- Vision cones will most likely not actually be cones, but rather collision boxes for ease of calculation.
- Cameras will be functionally the same as guards, but stationary. When cameras turn, they will also be in 90 degree increments.
- Lasers will just depend on the clock, no additional functionality required.
- A Level Manager will have to be created to load new levels.

Implementation Reflection:
We did walk cycles differently, as they are now instead an intrinsic part of a specific type of guard, specifically the patrol guards. However, otherwise we stuck to our plans for other implementations, such as the vision cones are really just long boxes as well as our “cameras” were replaced by a second type of guard, one that instead of moving just stands still and can look in every cardinal direction. Lasers do have a time functionality to them, as they turn off and on at different tunable intervals. Finally, our level manager was made in order to keep track of the levels and load them when necessary.

Distribution:

- Ketan: Walk cycle creation. vision cone/camera implementation, laser implementation. Music + SFX. Level design.
- Brandon: Sprite creation. Character controller. UI implementation. Level manager creation. Level design.

Distribution in Actuality:

- Ketan: Level Manager, Player controller, All level implementation, Obstacle and boundary implementation, level design
- Brandon: Sprite creation, keycard/door implementation, laser implementation, guard implementation, level design

Milestones (up to Alpha):

- End of Friday: Character controller, level one designed, one detection device implemented (camera, laser, etc).
- End of Saturday: Level manager created, additional detection device implemented, walk cycle created. Level two designed.
- End of Sunday: Complete level one implemented.

Dragonfly Engine Game Design Document

- End of Monday: Complete level two implemented (it will take much less time now that we have one working level) UI, SFX, bug fixing.

Milestones in Actuality:
We stuck to our original plan for milestones pretty well. We had differing amount done each day, some days we were further along than the milestones and some we were behind, but overall we mostly were able to get the work done in a timely manner and didn't put all of it off until the end.