Phase III

Reilly, Terrence Justice, James Goodall, Brad
Maresh, Keefer Gleason, Meagon

February 19, 2017

External Design Specifications

Software Architectural Design Specifications

Detailed Design Specifications

Interface Specifications

The classes `RenderedBase` and `ActorBase` will form a basis for the API for LazerBlast. They describe all moving objects, and all actors in the game. Among their most important functions, they will define a method to set the current action. They will also use Python's magic methods to get the next frame of their actions.

Class Definitions

- `RenderedBase` will implement the basic interface for any object which is rendered on the screen. It provides a way of getting the next image in an sequence for rendering.
- `ActorBase` will describe the interface for actors. All objects which are rendered to the screen and perform some action (the user and the enemies) will be subclasses of this class.

Pseudocode

```
# Terrence's section
class RenderedBase(object):
    images = dict()
    sounds = dict()
    _action = None
    _action_i = -1
    box = pygame.Rect(0, 0, 0, 0)

    def set_action(self, action):
        if action not in self.images:
            raise Exception
```

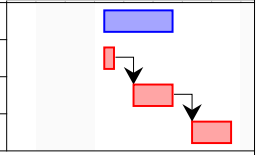
```
        self._action_i = 0
        self._action = action

    def __next__(self):
        if self._action_i == -1:
            raise Exception
        self._action_i += 1
        return self.images[self._action][self._action_i - 1]

    def render(self, context):
        pygame.draw.rect(context, (255, 0, 0), self.box)


class ActorBase(object):
    def __init__(self, health=0, weapons=list()):
        self.health = health
        self.weapons = weapons
        self._weapon_i = -1 if len(self.weapons) == 0 else 0

    def add_weapon(self, weapon):
        if self._weapon_i == -1:
            self._weapon_i = 0
        self.weapons.append(weapon)

    @property
    def weapon(self):
        if not (0 <= self._weapon_i < len(self.weapons)):
            raise Exception
        return self.weapons[self._weapon_i]

    def next_weapon(self):
        self._weapon_i = (self._weapon_i + 1) % len(self.weapons)
```

Data File Specifications

The data file for LazerBlast will consist of a pickled state of all high scores, and, possibly, of the current game state (if a user wants to continue a longer game.) The format itself is decided by Python, and will represent the internal python objects.

Test Plan Specifications

Appendix

		Name	Duration	Start	Finish	Predecessors	Resource Names	19 Feb 17 F S S M T W T F S
1		Menu Outline	3 days?	2/20/17 8:00 AM	2/22/17 5:00 PM			
2		Abstract Actors	1 day?	2/20/17 8:00 AM	2/20/17 5:00 PM			
3		Moving Actors	2 days?	2/21/17 8:00 AM	2/22/17 5:00 PM	2		
4		Actor Interactions	2 days	2/23/17 8:00 AM	2/24/17 5:00 PM	3		