**Helicopter Rescue 3, Cycle 1**

**Michael Orzel, Nicholas VonDollen, Taylor Cook, William Blair**

**Administrative Digest**

**SCHEDULES**

**Weekly Friday meetups.**

Our group followed one week schedules, which were decided through in class meetings and weekly Friday meetings. Each week we set a new goal and assignment for each member of our group.

**Friday Meetings:**

Date: 02/17/17

Start time: 7:00pm

End Time: 7:30pm

Total Time: 30 min

Description:

-Split responsibilities for cycle 1

Date: 02/24/17

Start Time: 7:00pm

End Time: 8:00PM

Total Time: 1hr

Description:

-Code review of BJ, Nick, and Taylor's Code

-Updated weekly report

(No meetings March 3rd or March 10th due to Spring Break)

**FORMAL PRESENTATIONS**

See attached powerpoint

**STATUS REPORTS**

See attached documents

**MEMORANDA**

Our goals for cycle 2 will be to create a more solid and playable Helicopter Rescue game. This would entail:

* Object/building collision
* Pause Screen
* Start Screen
* End of Level / Win Screen
* Level Balancing
  + Make the level completable but challenging through the spacing of objects and buildings
* Lives / Death
  + If the player loses all of their lives, they lose the level

**LESSONS LEARNED**

**Pros:**

When Mike couldn’t get the sdl\_image or sdl\_ttf to work, he coded through Nick. Mike had to send code to Nick to compile/test it, it worked out in the end.

Facebook worked well for communication

Access to each other’s branches for collaboration/git was useful

Group communication: everyone kept each other up to date/informed

**Cons:**

Integrating Code - all worked off of separate base code - not everyone had the most up to date code to work off of - more difficult to integrate

**Helicopter Rescue 3, Cycle 1**

**Michael Orzel, Nicholas VonDollen, Taylor Cook, William Blair**

**Software Development**

**VERSION DESCRIPTION**

Helicopter Rescue version 1.0

**Description:**

This version of Helicopter Rescue is a rough estimate of our proposed final product, containing elements that are the building blocks of a complete Helicopter Rescue. In this version, a helicopter is programmed using MiniAT by the player in assembly.

The player has the ability to move the helicopter through the application of thrust in both vertical directions, but has to compensate for the helicopter’s inertia and momentum. The level automatically moves forward for the player, but will eventually be able to force the level to move at a faster pace by accelerating the helicopter forward (this will be implemented in a future version).

Victims and obstacles are automatically generated at varying positions throughout the level, and a background automatically moves the level forward as well.

**Missing:**

Currently Helicopter Rescue is missing victim rescuing and obstacle collision

**Bugs:**

none known currently

**FEATURES**

**MVPP Feature #3**

**Name:** Helicopter movement

**Feature**: The programmer can apply ‘thrust’ to the helicopter in the cardinal directions (up, down, left, and right). The thrust has both separate vertical and horizontal values, with a range of -100 (for backwards) and 100 (for forwards). The helicopter will not move instantaneously or stop instantaneously as well. It will have inertia and momentum adding both realism and an extra challenge for the player.

**Constraints:** The helicopter can move any direction on the screen but cannot make the level move backwards. If the helicopter hits an object the player will either lose a life or lose the level. The helicopter hitting a victim results in that victim’s death and a deduction of points.

**MVPP Feature #6**

**Name:** Background scrolling

**Feature:** The background automatically scrolls independent of the helicopter.

**Constraints**: The background doesn’t scroll backwards. The helicopter cannot go off screen. Scroll speed is capped so it moves slowly.

**MVPP Feature #8**

**Name:** Victim Generation

**Feature:** Victims spawn throughout the entire map for the player to rescue using the helicopter’s movement feature illustrated in Feature #3.

**Constraints:** The victims should spawn between a level of 5000 x 768 resolution, and should not clip through the floor or the ceiling of the level, but may clip from the front or to the back, ensuring that victims spawn within reach of the player, while still providing the rush of a challenge. The victims should scroll from right to left throughout the level so that the player-controlled helicopter may miss them.

**MVPP Feature #9**

**Name:** Environment Generation

**Feature:** Generate objects for the entire level, and make the objects scroll (1 pixel per frame).

**Constraints:** Make sure objects are generated properly (not overlapping). Ensure objects are not scrolling too fast or too slow. Ensure helicopter has the ability to navigate through the objects (they aren't too close together). Ensure objects move at the same speed as victims.

**DESIGN**

* Score:
  + Increase for each victim rescued
  + Increase for each distance checkpoint
  + The faster the level is completed, the more points awarded
  + Decrease for each victim killed
* Collision:
  + Helicopter collides with obstacle: lose a life
  + Helicopter collides with a victim: if hit with helicopter, victim is killed
  + Helicopter collides with ceiling/floor of window: no problem as long as no obstacle is there
* Obstacles:
  + Obstacles are part of the level, levels are premade
* Rescuing Victims:
  + The helicopter drops a rope to reach the victim
  + The victim has to ‘climb’ the rope
  + While the victim is climbing the rope they are vulnerable to other obstacles
* Helicopter Physics:
  + The helicopter follows a set of physics rules (inertia/momentum), so it won’t start or stop moving instantly
* Helicopter Movement:
  + The helicopter will have a x and y velocity, which will be affected by thrust
  + The helicopter can move anywhere on the screen
  + Cannot move backwards through the level - Super Mario Bros level progression
  + The helicopter can touch each edge of the screen and not crash, as long as an object isn’t there
* MiniAT Movement:
  + Peripherals to apply horizontal and vertical thrust, range [-100, 100]
  + Peripherals to return the helicopter’s x and y position (in exact pixels)
* MiniAT Sensors:
  + Life sensor
    - Peripheral to return the number of lives the player has left
  + Victim sensor
    - Peripheral to return the horizontal distance to the nearest victim
    - Peripheral to return the vertical distance to the nearest victim
  + Object sensor
    - Peripheral to return the horizontal distance to the nearest object
    - Peripheral to return the vertical distance to the top of the nearest object

**TEST MATERIAL**

See attached documents

**SOURCE CODE**

See attached source code