Implementation

Group 8 Members

Zac Challis

Viktor Atta-Darkua

Dandi Harmanto

Harsh Mohan

Danny Burrows

Github Repository

Significant new features

- **Enemy ships shoot towards player** UR_SHIP_COMBAT
  The player is able to engage in combat with enemy ships as demonstrated in the tests : *MT_BULLET_DODGE*

  As both player controlled and AI enemy ships have health managed by the Pirate component; health is reduced with the takeDamage() method which allows the current health of a ship to be decremented by a specified amount. Enemy health bars were implemented to make combat more responsive and were displayed using a health bar renderable associated with the NPCShip class.

- **Capture of colleges by combat** UR_HOSTILE_COLLEGE_CAPTURE
  The player is able to choose to capture or destroy a college when all of the associated college's buildings are destroyed; this is demonstrated in the tests :*MT_CAPTURE_MENU* and *MT_CAPTURE_MENU_CHOICE*

  A new CaptureManager was required to handle the process of capturing/ destroying a college through the methods captureHandler() and destroyHandler(). When all buildings of the Building class are destroyed, PirateGame detects this and loads the new interactable capture menu, presenting the user with two buttons to capture or destroy, based on the input the appropriate method is called.

- **Player defeat in combat** UR_GAME_LOSE
  When the player's health is reduced to 0, the game loads the appropriate game over screen and exits; this is demonstrated in the test : *MT_GAME_LOSE*

  A new EndScreen is created by PirateGame and displayed when the user's health reaches 0. The text on the screen will always display a game over message unless the win() method is called upon the completion of all issued quests

- **Interactable obstacles should randomly spawn on the map**
  UR_OBSTACLE_ENCOUNTER
  Obstacles in the form of sea monsters and boulders are rendered across the world map. Upon collision with a sea monster player health should decrease and upon collision with a boulder player armour should decrease. This is demonstrated in the tests : *armorDamageOnBoulder, healthDamageOnMonster and MT_OBSTACLE_COLLISIONS*

  New entities Boulder and Monster were created and are rendered as rigid bodies, the setCallback() method extends CollisionCallBack ensuring obstacles are collidable. Obstacles can be created the appropriate GameManager methods: CreateBoulder() and CreateMonster()

- **Plunder should be able to be used to purchase power-ups**
  UR_SPEND_MONEY
  Players can press appropriate number buttons enabling them to expend earned plunder on power ups that aid their gameplay. This can be demonstrated in the tests : *ammo/ armour/ health/ immunity/ infiniteAmmo/speedHandlerPaid*

A shop renderable is constantly displayed as part of the HUD, despite not directly interacting with the player, it serves as an infographic referring to the corresponding powerup a player will receive upon number button press. EnhancementManager constantly monitors user input in an update() method. Number buttons can be pressed at any time and the power up will be applied to the user given subsequent plunder; this makes the purchase of power ups more responsive, enabling the user to purchase power ups such as infinite ammo or invincibility in tight situations

The GameScreen is also updated to take into account the various enhancements, it creates Label variables for each enhancement cost (e.g. healthTax, speedTax, etc…) and sets them to the values of their variables in EnhancementManager using the getTaxation(enhancement e) method from enhancementManager. It does this in the update() method so that the values are updated after the game difficulty is chosen. These labels are then displayed in the game screen, alongside images of the enhancements in entity form which the GameScreen also displays, to allow the player to know how much various enhancements cost.

- **The game can be saved and resumed after quitting** UR_SAVE
  When the player quits the game by pressing the Esc key, they are provided with the option to save the game state. When the game is opened once again the provided *Resume* button on the main menu should reload the game from the save game state. This is demonstrated in the tests : *saveColleges, savePlayerStats, saveShips, loadColleges, loadPlayerStats, loadShips, saveDifficulty and loadDifficulty*

  The game has a new Manager class, SaveManager. This Manager comes with 2 methods, save() and load(). save() records important current game information such as player health, armour, ammo, location,etc… and load() retrieves this information so that player can continue a previously saved game.

  The concrete architecture (fig 3.1.4 in arch1) shows that SaveManager has a relationship with one or more screens. In particular, the new screen/ UI class, PauseScreen, in which a ‘save’ button will trigger the save() method from SaveManager. Similarly The old MenuScreen has been updated with a new ‘Resume’ button, which will trigger the load() method from SaveManager.

- **Difficult can be adjusted, affecting factors like damage to player, etc…**UR_DIFFICULTY
  When the user opens a new game a difficulty screen is loaded posterior to the play button being interacted with. Three interactable buttons should display difficulties of *Easy, Medium* and *Hard,* which should load appropriate game settings upon interaction.

  This is demonstrated in the tests: *easy/medium/hardPlayerHandler, easy/medium/hardCaptureHandler, easy/medium/hardDestroyHandler, easy/medium/hardEnhancementHandler and easy/medium/hardEnhancementCostHandler*

  The Game has a new Manager class called DifficultyManager. The class has three core methods, easyHandler(), mediumHandler() and hardHandler(). On being used, the different difficulty handlers will:

  - Adjust player values such as starting health, armour, ammo, etc… using methods from Gamemanager

- Adjust the effect and cost of different power-ups using methods from EnhancementManager
- Adjust the gold and xp bonus of capturing and destroying colleges using methods from CaptureManager
- Adjust the damage dealt, bullet speed and gold and xp gain on destruction of enemy ships using methods from the Entity Ship

Starting a new game from the MenuScreen will take you to a new screen called LevelScreen. This screen has buttons for the 3 difficulties of easy, medium and hard, clicking the respective button will call the respective Handler method from DifficultyManager and set the difficulty by adjusting various parameters.

- **Player can receive power-ups, e.g. temporary infinite ammo, invincibility, etc…**UR_POWERUPS
  Powerups should be able to be purchased with appropriate plunder upon press of the correct number button. Powerups should also be rendered as collidable objects on the map. Powerups implemented provide : Increase of health/speed/armour, infinite ammo and invincibility

  This is demonstrated in the tests : *ammo/ armour/ health/ immunity/ infiniteAmmo/speedHandlerPaid* and *ammo/ armour/ health/ immunity/ infiniteAmmo/speedHandlerFree*

  The Game handles power-ups mainly by using the new Manager EnhancementManager. Variables such as health, ammo and armour affect how much effect the various power-ups will have on the player (e.g. if health is equal to 20, a health enhancement may increase the player's health by up to 20). Other variables such as healthTax, armorTax, ammoTax, etc… affect how much various power-ups cost if the player wants to buy them with plunder.

  As can be seen in the concrete architecture, the Enhancement entity has a relationship with EnhancementManager. On collision with the entities representing various enhancements on the map, the enhancement entity will alert the EnhancementHandler that the respective enhancement has been gained by calling the respective Handler method (e.g. healthHandler(`boolean free`), speedHandler(`boolean free`), etc...) from EnhancementManager and giving it a true boolean value as input.

Features not implemented

- **Bad weather objects appear on map to thwart the player**
  UR_WEATHER_ENCOUNTER
  We Implemented monsters to decrease nearby player health and boulders to decrease nearby player armour. Both are obstacles in the game that make it more interesting. Although bad weather is different from them in that you can steer through it, we did not pay much attention to it as it doesn't seem to add too much functionality when we already have boulders and monsters. In the end, It did not get implemented as we paid more attention to other issues.

- **Enemy colleges should shoot cannonball at the player**
  UR_HOSTILE_BUILDING_COMBAT
  Due to the player combat between multiple enemy ships, battlements were already too intense and proved difficult on simpler difficulties. College combat was removed in an effort to make gameplay more enjoyable.