EECS 448 Project 4 Documentation

Group 3

Authors:

Liam Connolly

Turner Graham

Carter Harrod

Dawson Kabler

Jarrett Zelif

Estimation of Person-Hours for Project Completion

I. Project 1 Person-Hours
    a. Meetings
        i. (6 meetings) * (5 persons) * (0.5 hours) = 15 person-hours
    b. Individual Coding
        i. (1 person) * (5 hours) = 5 person-hours
        ii. (1 person) * (1 hours) = 1 person-hours
        iii. (1 person) * (4 hours) = 4 person-hours
        iv. (1 person) * (4 hours) = 4 person-hours
        v. (1 person) * (3 hours) = 3 person-hours
        vi. Total Person-Hours = 17 person-hours
    c. Group Coding
        i. (1 session) * (5 persons) * (4 hours) = 20 person-hours
    d. Total Person-Hours
        i. 15 + 17 + 20 = 52 person-hours

II. Project 2 Person-Hours
    a. Meetings
        i. (4 meetings) * (5 persons) * (0.5 hours) = 10 person-hours
    b. Individual Coding
        i. (1 person) * (3 hours) = 3 person-hours
        ii. (1 person) * (2 hours) = 2 person-hours
        iii. (1 person) * (2 hours) = 2 person-hours
        iv. (1 person) * (4 hours) = 4 person-hours
        v. (1 person) * (2 hours) = 2 person-hours
        vi. Total Person-Hours = 13 person-hours
    c. Group Coding
        i. (2 sessions) * (5 persons) * (5 hours) = 50 person-hours
    d. Total Person-Hours
        i. 10 + 13 + 50 = 73 person-hours

Estimation of Person-Hours (Continued)

III. Project 3 Person-Hours

a. Meetings

i. (2 meetings) * (5 persons) * (0.5 hours) = 5 person-hours

b. Individual Coding

i. (1 person) * (2 hours) = 2 person-hours

ii. (1 person) * (1.5 hours) = 1.5 person-hours

iii. (1 person) * (3.5 hours) = 3.5 person-hours

iv. (1 person) * (3 hours) = 3 person-hours

v. (1 person) * (0.5 hours) = 0.5 person-hours

vi. Total Person-Hours = 10.5 person-hours

c. Group Coding

i. (1 session) * (5 persons) * (4 hours) = 20 person-hours

ii. (1 session) * (3 persons) * (3 hours) = 9 person-hours

d. Total Person-Hours

i. 5 + 10.5 + 20 + 9 = <u>44.5 person-hours</u>

Project 4 is going to involve minor updates and additions to Project 3 as well as the creation of a short video presentation. The minor updates and additions include better graphics, sounds, a scoring system, a main menu interface, and a better overall user interface. These should take little time to implement, as they are simple, and each group member has a firm grasp of coding in the Python language.

Considering the previous projects' person-hours for updates and additions, the projected person-hours to implement these enhancements is 25 person-hours. As for the video presentation, we have no information as to how long this will take from previous projects, as all three projects before this did not include any video presentation. However, two of our group members are familiar with video-editing. Their estimate for the filming and post-production of an 8-10 minute video presentation is 3 person-hours.

Taking these estimates, and factoring in a bit more time for any unforeseen errors and challenges, a good estimate for the total person-hours required for Project 4 would be 30 person-hours.

Accounting of Actual Person-Hours

I. Turner Graham
   a. 11/7/20: 3
   b. 11/8/20: 5

II. Dawson Kabler
   a. 11/7/20: 1
   b. 11/8/20: 2

III. Liam Connolly
   a. 11/7/20: 3
   b. 11/8/20: 5

IV. Jarrett Zelif
   a. 11/7/20: 3
   b. 11/8/20: 3

V. Carter Harrod
   a. 11/7/20: 3
   b. 11/8/20: 5

VI. Total Person-Hours
   a. 33 hours

Defect Tracking

Date Reported	Reporter	Description of Defect	Date Fixed	Fixer	Description of Fix
11/7/20	Liam	Bullet's graphics same as the Player's	11/7/20	Liam	Moved Ship png file to correct class
11/7/20	Jarrett	Ship graphic the size of the entire window	11/7/20	Liam	Resized Ship png file
11/7/20	Carter	Score outputting incorrect value	11/7/20	Carter	Rewrote the function calculating score
11/7/20	Turner	Alien hitbox too small	11/7/20	Jarrett	Resize hitbox dimensions of Alien
11/7/20	Carter	Writing to high score file yields incorrect format	11/7/20	Carter	Switch from concatenate to append function for file writing
11/8/20	Carter	Issues with developing test bench for game	11/8/20	Carter	Moved variables to global
11/8/20	Turner	Ship not shooting from center after implementing sprite	11/8/20	Dawson	Increased the offset
11/8/20	Dawson	Software for documentation formatting issues	11/8/20	Dawson	Rearranged if statement

Process of Integrating Code from Team Members

For Project 4, our group had a series of updates and additions to Project 3 that we wished to implement. Each member of our group worked on a particular addition or update to the software individually. We coordinated with other teammates when certain additions overlapped in their functionality or when we needed assistance in our implementation. Each member worked on their specific addition until they arrived at a working product. Once each of our individual implementations were complete, we came together to integrate the additions into one uniform, functional implementation of Project 4. Our strategy of code integration matches that of the "implement everything, then integrate" approach.

Normally, this approach to code integration not desirable as artifacts that call other artifacts cannot be tested separately. This would have been an issue had our desired additions to the code been highly dependent on one another. However, our additions were quite independent from one another. For example, the game sounds do not depend on the level tracker for its functionality, and vice versa. Additionally, the high score reading/writing does not depend on the new graphics for its functionality, and vice versa.

Despite our integration strategy's tendency to reveal major design flaws late in the integration process, no such catastrophic errors were found. During the integration process, we encountered few errors as each addition to the project worked independently of the other additions, as mentioned above. The errors that were found were due to incorrect syntax and variable naming. These errors were easily fixed, making the integration process go along quickly and without much trouble.

Deployment Plan

Because of the nature of our project being a game, there are many options as to where we can deploy it. This game is a computer game that would not be able to work on mobile devices thus removing any mobile app stores like Apple's App Store or Google's Google Play Store as options. This is why we believe that deploying our game on a computer platform such as Steam is our best option.

There are many steps required in getting our project deployed on Steam's platform. First thing that would need to be done is a plethora of paperwork. In this paperwork we will need payment information such as a routing number, bank account number, and bank address so that Steam knows where to send payments from sales of our game. We would also have to determine our tax status and withholding rates where we would register our group of developers under the company name "Bunch O Fun" and we would fill out the appropriate tax forms. Along with our payment and tax information, we would need to submit a "Product Submission Fee" in order to get fully set us. We will pay a total price of $100 for this fee.

After getting through all of the paperwork we will need to access "Steamworks" and prepare our game for release. In this part of the process, we will build a store page, upload the game, and add any extra features along with setting the price for the game. We would also have to submit documentation in Steam's required format.

Before the store page can go live, there is a brief review process where Steam runs our game and checks out our store page. This is to make sure that everything is configured correctly and runs as expected without doing anything harmful. This process takes around 1 to 5 business days. Before the game is finally released, there is a 30 day waiting period where Steam reviews our information to make sure they are okay to do business with us. During this waiting period, we can set up a "coming soon" page to build interest in our game to our audience. After this, we are free to upload and update the game as we see fit.

The potential market for this game is any person who is interested in video games. This because "Space Invaders" is such a classic game, this game would appeal to anyone who either wants to have a few minutes to kill or if they are a more serious gamer and would like to get the highest score.

As for the costs of the product, it is pretty straightforward. The cost to get our game on Steam is $100 and the rest of the costs would be towards whatever sort of marketing we might be interested in. This could include setting up demonstrations of our game at gaming conferences or at developer conferences hosted by Steam or other gaming communities. Overall, this is a rather affordable product to deploy.

Maintenance Plan

There are four types of software maintenance: perfective maintenance, adaptive maintenance, corrective maintenance, and preventive maintenance. Perfective maintenance involves functional changes that are new or based on user requirements and enhancements to increase the system's performance. Adaptive maintenance involves an environment change for the software and changes in hardware, software, operating system, or organizational policy. Corrective maintenance involves fixing bugs, errors, and faults in software that has already been released and is being used by users. Preventive maintenance involves changes that ensure the longevity of the software such as code optimizing, code restructuring, and documentation. Perfective maintenance, adaptive maintenance, corrective maintenance, and preventive maintenance account for 50%, 25%, 21%, and 4%, respectively, of the costs of software maintenance.

As our product is fairly small in its scale and simple in its implementation, we would need relatively few junior software developers to maintain our game, perhaps two or three. The average salary for a junior developer in the United States is $82,000, meaning we would pay roughly $6,800 per developer per month. Let's say we payed three junior developers to maintain our software, meaning our cost for developers would total $20,400 for one month of maintenance.

Here we will mention some of the additional costs associated with the maintenance of our software for a month. Putting our game on Steam has a $100 publishing fee associated with it. If we had a website associated with our game, hosting that site would cost between a few dollars up to several hundred dollars a month. Buying a decent hosting service, we would spend about $20 a month for a website. As our software is a single player game, we would not need to pay for hosting servers.