Problem 1.1.7

Scratch Game or Story

Introduction

Software developers sometimes use their skills to create a product for a client. At other times, developers create a product for themselves. Whether meeting their own needs or the needs of a client, collaboration allows developers to work on bigger projects and have more fun.

The way that team approaches a problem is called their methodology. In this problem, you will apply the Agile methodology.

What do you want to create? How will you work with others to define the problem and create the solution?

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

Materials

- Computer with Scratch™, Internet access, and camera
- Project notebook

Resources

Problem 1.7 Rubric

1.1.7 sourceFiles.zip

Procedure

1. Form pairs as directed by your teacher. Meet or greet each other to practice professional skills. Set team norms.
2. Review the Game and Story Criteria for this project. You will choose either a game *or* a story to develop.

Game Criteria

- User interaction

  The user should be able to use keyboard and/or mouse input in a way that fundamentally affects what happens.

- Objective

  The game should have an objective with several or as many degree of progress toward the objective as possible. A score would be sufficient, but many alternatives exist.

- Multiple states

  The game should include different states in order for the user to experience variety. Levels during which the difficulty changes or bonus stages appear would be sufficient, but many alternatives exist.

Story Criteria

-
-
  - Multiple acts

    The story should have different sections, like multiple acts in a play. For example, the story might occur on different stages, but many alternatives exist.

  - User interaction can affect story line

    The user should be able to use keyboard and/or mouse input in a way that fundamentally affects what happens.

  - User interaction between story line branch points

    The user should be able to use keyboard and/or mouse input in a way that controls what is occurring within at least one of the acts.

  - The project at http://scratch.mit.edu/projects/12586146/ can be used as an example for structuring a story with a state map.

3. Review the Project Rubric.

4. Brainstorm ideas using tag lines and thumbnail sketches. Follow the guidelines for brainstorming: never criticize ideas during brainstorming, but “piling on” is welcome.

5. Develop one or two of your ideas with further discussion and documentation.

6. Decide on one game or story that you will develop into a product. Using diagrams, lists, and flowcharts, explain to another team of developers what you want the product to do. Up to now, you have been (mostly) playing the role of the client. Now you will transition to the role of developers. When another team tells you what they want their product to do, pretend they are your client and you will create their product. If they are not giving you enough information to create the product to their satisfaction, ask questions.

7. You will now serve as the software developer for your own project. Plan the first sprint. The planning session is normally **timeboxed**, meaning it is limited to 3 minutes of planning for each hour of the sprint. Since this is your first sprint, you might take longer.

   Your plan should include two bullet point lists, as follows:

   - A product backlog provides the most important user stories at the top. User stories lower on the list can be large and poorly defined.
   - A sprinttask list. Sprint backlog items should be specific and broken into small tasks.

8. Strategize, code, and test in small increments.

   - Switch driver and navigator roles every 10 minutes or so.
   - Include Scratch comment balloons as you develop your solution.

9. Write a reflection on how well your product meets the clients' needs. Include descriptions on strengths and weaknesses of the solution as well as potential plans for the next sprint.

10. Practice Opportunity for the *Create* Performance Task

    “Create a 50-59 second video in which you demonstrate the running of at least one significant feature of your program.” (Adapted from College Board Create Performance Task Part 1.)

11. Practice Opportunity for the *Create* Performance Task

    “Identify the purpose of your program and explain what the video illustrates. (Approximately 150 words)” (Adapted from Create Performance Task Part 2a.)

12. Practice Opportunity for the *Create* Performance Task

    “Describe the incremental and iterative development process you used, focusing on two distinct points in that process. Describe the difficulties and/or opportunities you encountered and how they were resolved or incorporated." (Adapted from Create Performance Task Part 2b.)

13. Practice Opportunity for the *Create* Performance Task

    Your entire Scratch program is an algorithm to repeatedly look for events and execute an event handler for each user or program event that occurs and has a handler. Your algorithm for the event loop combines the algorithms of each event handler that you created. "Describe how each algorithm within your algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program. (Approximately 200 words)” (adapted from College Board *Create* Performance Task Part 2c.)

Note: This last four deliverables are adapted from the official College Board Create Performance Task but they do not duplicate the content of College Board Task or Rubric. The task provided here contains elements that are different than the College Board Performance Task and Rubric. Please reference official College Board materials.

Conclusion

1. Reflect on the creative process you used. What was useful? Discuss your reflection with your partner and then write a reflection individually.
2. Reflect on the team dynamic. What helped the team work well together? Discuss your reflection with your partner and then write a reflection individually.

Problem 1.1.7 Scratch Game or Story Rubric

	4	3	2	1
Solves Problem	Artifact fully addresses personal, practical, or societal intent posed by problem statement	Artifact addresses the personal, practical, or societal intent posed by problem statement	Artifact mostly addresses the personal, practical, or societal intent posed by problem statement	Artifact does not adequately address the personal, practical, or societal intent posed by problem statement
Documentation	Uses appropriate documentation of work.The three formats for documenting work: • Scratch comments • Project Design Notebook • Named versions of project	Uses appropriate techniques in 2 forms for documenting work	Often uses appropriate techniques for documenting work	Does not usually use appropriate techniques for documenting work
Collaboration	Provides helpful original input to others Promotes positive, productive, and respectful team dynamic Encourages and incorporates input from others Promotes equitable workload	Provides adequate original input to others Maintains positive, productive, and respectful team dynamic Positively incorporates input from others Maintains equitable workload	Significant but limited input Usually maintains positive, productive, and respectful team dynamic Receives input from others Shares workload somewhat equitably	Limited input Is not promoting positive, respectful, or productive team dynamic Discourages or is unresponsive to input from others Does not promote equitable workload
Other comments:				
Presentation	Effective presentation techniques: Posture Gestures	Mostly effective presentation techniques: Posture	Mostly adequate presentation techniques: Posture Gestures	Inadequate presentation techniques: Posture Gestures

	Voice Eye Contact	Gestures Voice Eye Contact	Voice Eye Contact	Voice Eye Contact
Appropriate Algorithm	Code demonstrates use of appropriate algorithms	Code mostly uses appropriate algorithms	Code often uses appropriate algorithms	Code does not use appropriate algorithms
Explanation of Algorithm	Documentation (comments) clearly and thoroughly explains the algorithm(s)	Documentation explains the algorithm(s)	Documentation insufficiently explains algorithm	No documentation
Explanation of Problem Solution	Prose clearly and thoroughly explain how the solution meets the need Prose clearly explains the solution's strengths and weaknesses and strategizes for improvement	Prose explains how the solution meets the need Prose mentions a strength or weakness and ideas for improvement	Prose explains how the solution meets the need Prose mentions a strength or weakness	Prose does not address how the solution is connected to the need
Planning	Product backlog shows intent Sprint task list subdivides to simpler subproblems	Product backlog shows intent Sprint task list shows subdivision to simpler subproblems, but some tasks are too big	Product backlog and sprint task list show inadequate attempt to indicate long-term intent and an attempt to break down problem into simpler tasks	Product backlog or sprint task list are missing.