Introduction to C++

|  |  |
| --- | --- |
| **Assessment Task Number:** Part 3 – Document Version Control Use | |
| **Unit Code(s):** | **Unit Title(s):** |
| ICTPRG443 | Apply intermediate programming skills in different languages |
| ICTICT449 | Use version control systems in development environments |
| **Instructions to Learners:** | |

This subject covers the use of version control across multiple topics. Each topic contains a set of exercises that will guide you through creating a repository and performing common actions, such as committing, pushing, pulling, and branching.

It is recommended you complete all version control exercises as you complete the requirements of this assessment task.

**You must make at least three commits to your project**. You may want to do this as you work the version control exercises.

All documentation may be contained within the one MS Word or PDF document.

All URL links must be accompanied by a screenshot that shows the same URL. This ensures that your evidence can be assessed and archived even if your Git repository is removed or becomes inaccessible.

**Document Version Control Use:**

Every tool we use should solve a problem with our work or the way we work together.

Create a brief document that provides the following information:

1. A link to your GitHub repository
2. A link to the merge commit you created
3. A link to the pull request you filed as a part of your Version Control Exercises
4. A problem statement describing why we use Git for version control:
   1. What is the problem we are solving?
      1. Who is experiencing the problem?
      2. What are our requirements for a possible solution?
   2. How does this solution (Git) solve this problem?
      1. What are its advantages?
      2. What are its disadvantages?
5. A list of resources you referenced to when learning Git (besides AIE resources).

**Problem Statements**

This information will help guide your response to item four (4) above.

Problem statements summarize key information about the problem and the environment surrounding it. They are often brief, but can provide detail as necessary to describe the problem.

Consider this sample problem statement written for game engines:

*Small development teams may find it expensive to split their resources between building the game engine vs. building the actual game itself. Teams may need to dedicate at least 1-2 full-time employees (FTEs) to developing the engine as requirements change over the product lifecycle, increasing operational costs.*

*Developers may choose to use game engines middleware like Unity or Unreal Engine in exchange for a small upfront cost in the form of licenses or royalties. Engineers can be instead be dedicated to developing the game instead, requiring less engineers and thus reducing operational costs.*

*Game engine middleware is often maintained by teams with dedicated R&D teams, providing its users with the latest innovations in the games industry at a fraction of the cost. Teams may hire for users experienced in specific game engines, minimizing resources spent on training and ramp-up.*

*If a game engine does not providing the features required for a particular game, its cost savings are reduced. Furthermore, missing features or bugs with the feature set can be difficult to fix if source access is not providing, possibly making the game engine more expensive to work with than a custom built one.*

In a few sentences, the passage of text describe the following:

* **What is the problem?** – The cost of building and maintaining custom game engines
* **Who is experiencing the problem?** – Small or new game development teams
* **What is needed in a solution to this problem?** – Minimizes the game engine development costs

The solution, game engines middleware like Unity or Unreal Engine, are described as follows:

* **How does Unity or Unreal solve this problem?** – Licenses or royalties for the game engine can be much more affordable than building a custom engine
* **What are its advantages?**
  + Externalized cost of R&D teams
  + New hires may require less training and on-boarding to be effective due to common knowledge of the game engine
* **What are its disadvantages?**
  + Missing or incomplete features can require teams to devote some or all of the cost of developing those features in-house regardless

Bugs in the game engine may not be readily diagnosed fixed if source-access is not provided

|  |  |  |
| --- | --- | --- |
| **Task** | | **Evidence Criteria** |
| 1. | Link to GitHub Repository and Screenshot | A link and screenshot to a version-controlled project in GitHub.  The project must have at least 3 commits |
| 2. | Link to Merge Commit and Screenshot | A link and screenshot to a merge commit you created |
| 3. | Link to Pull Request and Screenshot | A link and screenshot to a pull request you filed |
| 4. | A Problem Statement | A problem statement describing   * What is the problem version control solves?   + Who is experiencing the problem?   + What are our requirements for a possible solution? * How does this solution (Git) solve this problem?   + What are its advantages?  1. What are its disadvantages? |
| 5. | Resource List | A list of resources you referenced when learning Git (besides AIE resources) |
| **Submission Requirements:** | | |
| You will need to submit the following:  A document in MS Word or PDF format containing your answers for all items (1-5) | | |

1. Answers

* A link to your GitHub repository

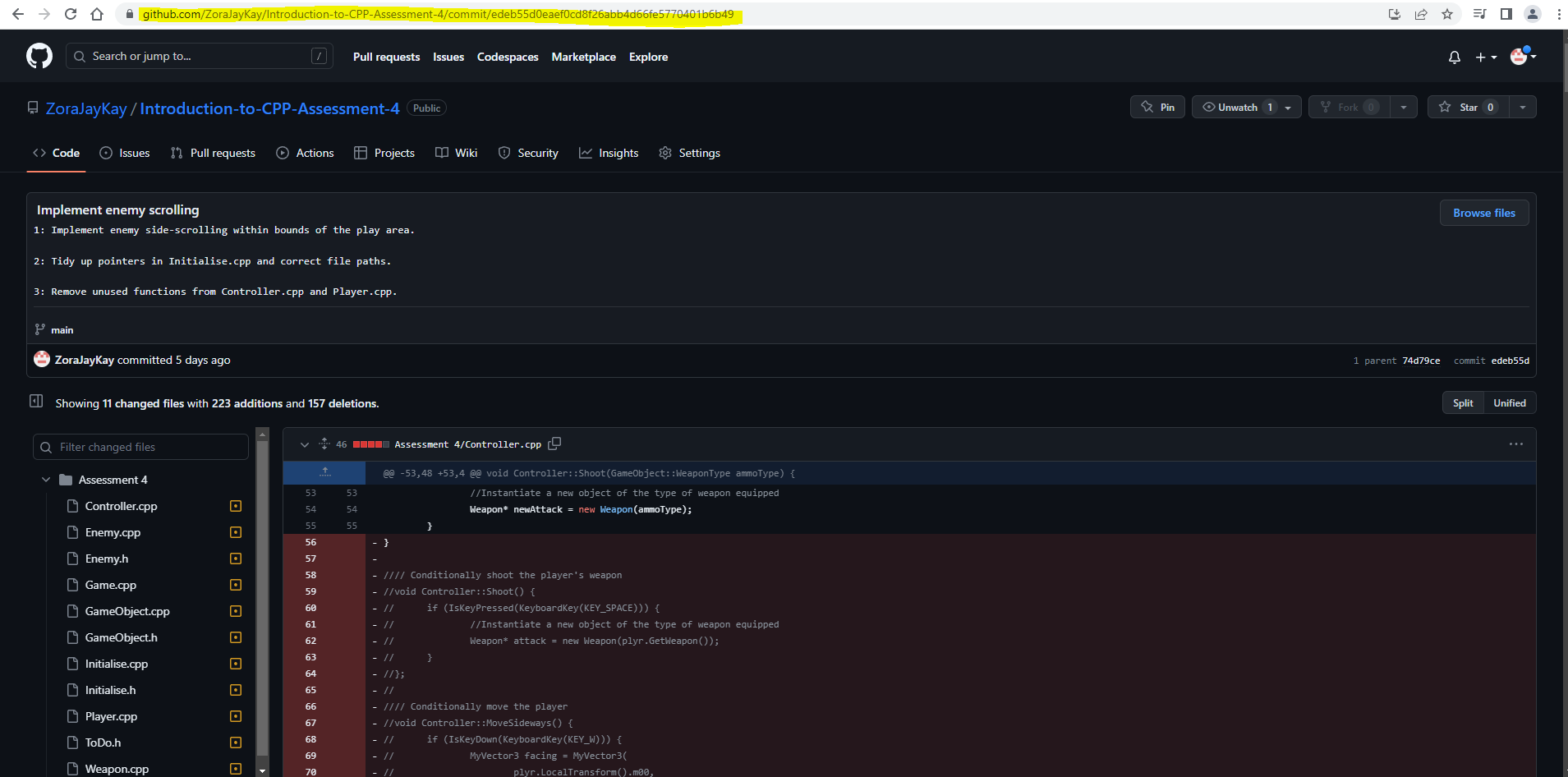
<https://github.com/ZoraJayKay/Introduction-to-CPP-Assessment-4.git>

A screenshot of a computer

Description automatically generated

* A link to the merge commit you created

<https://github.com/ZoraJayKay/Introduction-to-CPP-Assessment-4/commit/edeb55d0eaef0cd8f26abb4d66fe5770401b6b49>



* A link to the pull request you filed as a part of your Version Control Exercises

[Perform a branch split by ZoraJayKay · Pull Request #2 · ZoraJayKay/Introduction-to-CPP-Assessment-4 (github.com)](https://github.com/ZoraJayKay/Introduction-to-CPP-Assessment-4/pull/2)

A screenshot of a computer

Description automatically generated with medium confidence

* A problem statement describing why we use Git for version control:

*We use Git for version control because specific problems are guaranteed to arise without the use of a tool for reconciling different copies of a single thing, like a computer program or file.*

*The problem is that at a given moment in time there may be multiple versions of a single product which differ from one another. By using Git we are seeking to solve the problem of how to aggregate these multiple versions of the same product back into a single cohesive version. Separate versions of the same product may be complementary, such as where differences between versions do not overlap in any way, however versions may also conflict with one another, such as where two or more versions include the same information, the truth of which is contested. If a product permits the existence of more than a single unique copy of itself, any person, group of people, or other entity who works or relies on that product may encounter version control problems.*

*The solution to a product having multiple versions at any point in time is to:*

1. *Maintain a record of the contents of both the master product and the replica products at each point in time where version changes occur.*
2. *Implement a discrete process for every activity that records changes to the contents of a replica or master version, such as when a replica is taken, when changes are made to it, when changes are submitted for implementation back in the master, when diverging versions separate from one another rather than one overriding the other, etc.*
3. *Permit both the master and replica to revert their contents at a point in time to any other recorded point in time.*

*Git is one of many applications which implements this solution with each of these functions. The advantage of this solution is an exhaustive record of the moment in both time and version where conflicts between versions arise. One version can overwrite another, previously discarded versions can be recalled, comparisons between versions can be easily made, and specific conflicts can be isolated not just to specific conflicting versions, but to specific elements of their data, such as specific files, or specific elements / locations within those files (such as the characters within rows of code for Visual Studio).*

*The disadvantage to a version control system like Git is that in order for it to effectively mitigate the risks of version discrepancies, a minimum level of time and energy must be continuously invested to keep the version control system effective. Diligent record-keeping must be in place for all users of the system and all replica versions of the master file, including but not limited to adequate communication and descriptive use of language when documenting the nature, timing and extent of changes implemented, for reference by future users of the system.*

* A list of resources you referenced to when learning Git (besides AIE resources).

<https://docs.github.com/en/desktop/contributing-and-collaborating-using-github-desktop/working-with-your-remote-repository-on-github-or-github-enterprise/creating-an-issue-or-pull-request-from-github-desktop#creating-a-pull-request>

<https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>