9/10

Initial Meeting, Sprint 1

* Selected Zelda dungeon as project
* Agreed to use GitHub for version control – Ben will give GitKraken overview
* Tentatively picked Trello for project board
* Agreed to make functionality and whatever the rubric weights the highest the number one priorities in a pinch
* Agreed to keep meetings limited to as-needed and who needs to be there; not strictly necessary for all members to be present for every meeting as we have meeting notes, those who have the most relevant tasks should lead the charge
* Agreed to set an early deadline for sprints as a self-imposed check-in: if someone hasn’t responded or provided updates on the state of their work by the deadline, they surrender their right to do the work and other people may step in
* Utilize pull request system on Git to keep development contained and facilitate peer review – at least one reviewer for a pull request, ideally one more as a functionality tester
* Most of the group has limited experience with C# so Comer is our team expert, agreed to limit the use of **var** keyword as it introduces ambiguity
* Consider using Calvin’s Sprint 0 as a baseline – may be that it’s better to start from scratch
* Follow C# style convention and utilize resharper to create uniform code

Certified: Ben Horstman, Jacob Uligian, William Comer

9/12

Sprint 2 Task Planning

* Git overview
* Super-Update – abstract classes
* Utilize command pattern
* Delegated tasks – Each person will create the tasks on the task board for their larger work section
  + Ben – Player
  + Tyler and Calvin – Sprite Sheet and Enemies
  + William – Environment
  + Jacob – Items

10/4

Sprint 2 Reflection

The team did a good job planning ahead and dividing responsibility, as well as helping and redistributing work when necessary. Existing practices should be continued. The pull request system of documentation worked well, as it simplified documentation and put the impetus of contribution on each member; it will be continued. The Trello task board was effective for task tracking but should be used more frequently in future sprints. Part of the initial difficulty is due to the learning curve of the tool and use should be more apparent in the future.

Most of the difficulty encountered revolved around using new project management tools, as not all group members had prior experience with GitHub and Trello. These issues should abate as the team gains more experience with these tools. It was noted that some work was not completed until the sprint due date; no major repercussions occurred, but care should be taken to finish work earlier in the next sprints to allow more time for documentation.

Looking ahead: merge conflicts are likely to become more common in future sprints as tasks become more interconnected. The team should be sure to communicate about the state of the repository and work together to resolve difficult merge conflicts.

Shoutout: Ben did a great job on the player class and taking a leadership role.

Sprint 3 Task Planning

Level loading: Calvin, Tyler, William

Collision: Jacob, Ben

Considered adding GamePad support as a stretch goal. Decided to refactor enemy projectile generation to an independent class. Noted that environment tiles currently share most functionality; may need to be refactored to permit differences in collision checking.

Task assignments:

Tyler – build rooms and generate background sprites

Calvin – build parser

William – room manager / object generator  
Jacob – individual class collision

Ben – generic CollisionHandler

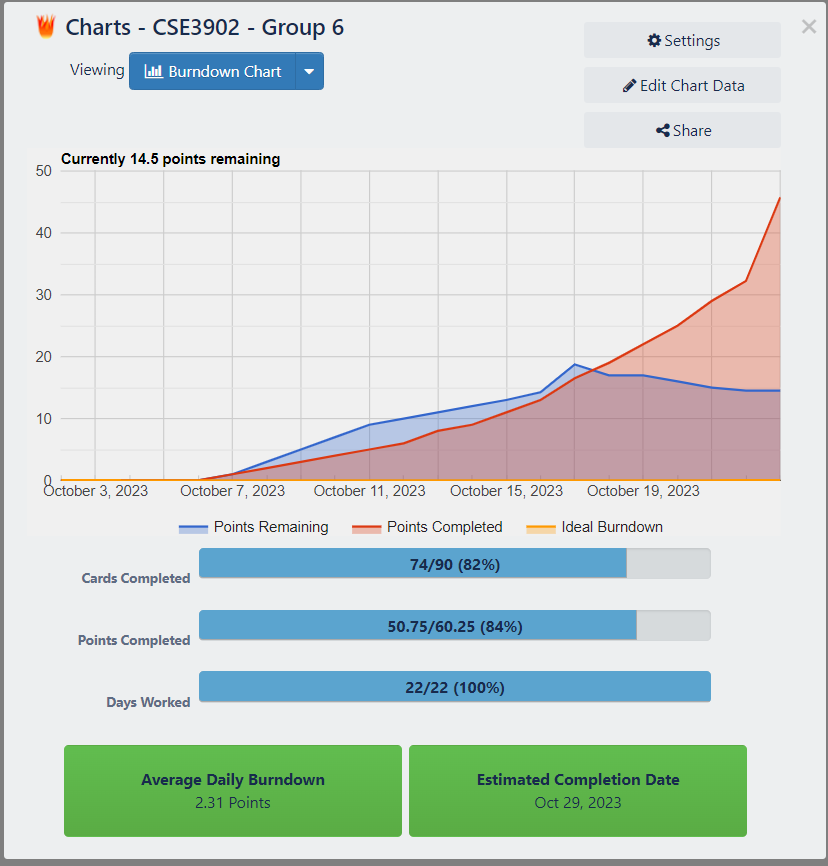
Code Review 10/18

* good job managing complexity
* private constructor for singletons - static handle collisions
* consider consolidating looping in collision classes as we have many helpers
* iboomerang in projectile collide w blocks
* remove all enemy manager
* double check coupling
* make sure to remove sprint2 old stuff
* textureatlassprite
* single sprite class is good
* refactor sprite selection

10/23 - Sprint 3 Reflection

Overall, the division of the sprint requirements was well managed. The addition of the burndown chart in Trello worked wonderfully. One thing to improve on moving forward is having a more concrete expectation of time to spend on each task; the “points yet to complete” part of the burndown chart increased significantly over the course of the sprint while it should have only gone down. Bug detection systems worked well; the combination of pull request reviews and manual testing as each member tries to use previously implemented items quickly identified logical errors and allowed us to nip several problems in the bud. Documentation improved as a whole, building on last sprint’s setup.

As expected, the primary sticking point of the sprint was merge conflicts. A lack of allocated time for resolving conflicts caused us to effectively miss the functionality check-in and delayed progress on the final submission. In the future, the team should have better communication about major refactors to ensure as limited a scope as possible.



Looking at code analysis, the cyclomatic complexity of the program doubled from the previous sprint (largely due to collision checking and object spawning switches) and class coupling increased by 50%. However, each of these metrics decreased for most individual classes, indicating that overall file/class structure has improved even as the project has gotten larger. A few notable weaknesses were Game1’s Initialize, as it still contains Command registration; the square collision checker, as it is hyperspecific to certain edge cases; and several HandleCollisions methods, as they rely heavily on switching. These should each be addressed and optimized in future sprints.

Looking ahead: HUD implementation will likely require a refactor of our current object position logic, care should be taken to allocate enough time for that.

11/13 – Code Review

This review helped to point out changes the team should make as judged by an outside perspective. There were many good choices such as a sound effect factory which made any excess of strings contained inside only the factory so that outside calls rely only on functions, or the GameObjectManager being a template is a unique but useful decision that other teams in the past have not tried. The code itself is very robust, almost to its own detriment because on the one hand it makes it very easy to expand for more functionality, but that comes at the cost of some readability and ease-of-use. A notable change that should happen is that the keyboard bindings that are created in the game1 class should be moved to an external class dedicated to creating bindings. Some other changes to make were some renaming, such as changing SharedTypes to something such as LevelLoadingTypes and changing the blade trap’s detector to something like “tripwire.” Another point was some areas that were hard-coded should be changed such as the PauseHud int literals or blade trap’s integer states; both would benefit from being changed to better containers like collections and enumerations. Single line comments should also be removed.

11/13 – Sprint 4 Reflection

For this sprint, the burndown chart was a much better representation of how the group’s time was used as the members got more used to working with the system and getting more active in using Trello as a whole. However, this sprint still has merging issues because of how the responsibilities were split up; merging the work made it clear that the interaction between pieces needed extra work to integrate smoothly. Issues from previous sprints persisted such as a lack of clarity in what exactly was causing merging issues. Although team members were effective in attacking merging bugs, a multitude of issues that were hard to find were caused by merges not interacting correctly.

The code analysis shows that the cyclomatic complexity and the class coupling each only increased by roughly 10% which is much lower of an increase than previous sprints. Certain individual items had very slight increases in each field; the overall stats increased due to additions such as new abstract classes, which streamlines coding from a development perspective.

Looking ahead: the project’s systems are set up to enable easy additions of new content that works in similar ways as current content. Expected issues lie with potential refactoring of the already complex systems to add new, possibly even simple, functionality that did not originally belong to the project.

A screenshot of a graph

Description automatically generated

12/6 – Sprint 5/Final Reflection

A screenshot of a computer

Description automatically generated

The team took a different approach to Sprint 5 than the other sprints: due to schedule constraints and the open-ended nature of the sprint, team members worked mostly independently throughout the sprint rather than having multiple people work on the same functionality. This worked surprisingly well but faced a few expected difficulties with merge conflicts. On the whole, the approach worked thanks to the effort that the team had made to make the code base modular and extendable throughout the project. Each additional feature added for Sprint 5 was simple to create and include in this project thanks to certain past design choices, such as using an abstract class in between several of our interfaces and concrete classes. Merging was slightly easier than previous sprints as no two group members needed to work on overlapping files.

Code metric analysis shows the expected level of increase in cyclomatic complexity and decrease in maintainability for the final sprint; much of this change can be attributed to the addition of multiple new Players and a new Dungeon, which act as a multiplicative factor for such attributes. The decision to implement additional player characters as Player States also negatively impacted code metrics, as those states are landmark low-maintainability files; a refactor to improve those would be recommended for future work.

The general opinion of the team is that the project went well overall and that its success was especially visible during Sprint 5. Over the course of the project, the team’s ability to work as a unit and utilize the required tools drastically improved and project organization/management stayed at a high level. Though code maintainability decreased over the project, as evidenced by the code metrics, the team believes that code quality largely did not suffer and that metric decreases were due to the inherent level of complexity of a project of this size. This is supported by the efficiency of Sprint 5; each team member reports that adding new content that fit within the developed framework was as easy as possible and that each segment of the project appeared to be easily extensible, which fits the project goals. Overall, the project was a success, and each team member is leaving with valuable group work experience and a nice gem to add to our programming history.