Zelda-Game

This Repository is for Team Alpha Won in CSE 3902 recreating the first dungeon from the original Legend of Zelda Game. This project will take four sprints to complete to add all the functionality.

V4

Controls

Keyboard

Controls	Function
Arrow Keys	to move Link around the screen.
A	to for Link to use the sword.
B	to use Items, Once paused used to select item. Keep clicking until the desired weapon is reached.
Q	to quit
R	to Reset
P	to Pause
H	to Randomize RoomData

GamePad

Controls	Function
Left Thumbstick	to move Link around the screen.
A	to for Link to use the sword.
X	to use Items, Once paused used to select item
Y	to quit
B	to Reset
Start	to Pause
Right Stick	to Randomize RoomData

Bugs

  The project is 95% Complete based on the functionality of the original game, but currently has 3 additional features implemented. Based game functionality currently has some bugs. The first bug being you can use multiple items at the same time ex. shoot 3 arrows in succession. Another bug is that when the enemies die, they despawn to quickly and as a result the death cloud can not be been. The Clock item currently only freezes the enemies and does not unfreeze them. The damage sound is played to many times when the player takes damage and can cause the game to crash in trapped in a corner. The pushable block moves many spaces if you want instead of just one space at a time. While it is possible to move left and right on the ladder, there is no where to go. The boomerang and the hand master cause you give/take damage instead of stunning. The spike trap does not have the intended functionality. One bug/feature is the sword is

one hit kill. This is because the animation is long and thus does damage the whole durations, while it makes killing things easy it is nice as the bosses are actaully hard so it gives a way to actually beat it

Bug Fixes
Links damage state got overhauled. Now there are four damage states in four directions. This fixes the bug where link is always looking right after taking damage instead he is looking the intended direction. Link can also not in the damage state deal damage to enemies via his sword or items. The player takes a little knock back to try and make sure link does not get trapped. Background music was turned back on. Compass not shows where the triforce pieces is with or without the map. The block in the compass room was moved back to original location. Link now has a function health system, including getting hearts, taking damage and displaying the hearts. Throwing items no-longer linger when passing room to room. The block edges seem to have been fixed for the most part, still some areas with certain movements can cause link to get caught on corners.

Additional Features
Our team currently has three additional features. The first feature is the introduction of the gamepad class which allows for the user to also use a controller for the gameplay, right not the keys are mapped for a xbox controller so the compatibility is very small. We also added a new boss that has two stages, and this 15 health. The boss is the one that drops the triforce piece now to win/beat the game. The last feature implemented currently is the randomization of the blocks and enemies in rooms. With a click of a button the rooms become chaotic, but the game is still playable as the items remain the same and thus allow you to still get the triforce piece. A second player was implemnted but with a visual studio crash and unltimate corruption of files they data could not be recovered and thus the feature was abandoned.

Code refactoring
Ankit ended up doing most of the code style and refactoring for the sprint to bring the code up to code quality standards. The first issue addressed was first found in sprint two where the textures of many classes were public, and they have since been changed to private readonly to prevent accidental changes. The next fix was with the item collection and to reduce the coupling all the logic on what the player has collected is not in the inventory class. This reduced the readability of the inventory class, but the inventory class was refactored to as many properties and methods became useless. The playerdoorloop class was very complex and had high cyclomtic complexity. This was fixed by creating static classes in separate files to handle the actual logic. The readability of the playerdooploop class increased drastically and the coupling went down as well. The static classes since handling only one function have high cohesion and thus also have high readability has they do the same thing. Game1 class coupling and cohesion was improved this sprint. Many of the coupling issues to game1 have been delegated to different managers. This includes the RoomManager that handles which room is present, the switching and the logic of updating and drawing them. The GameStateManager deals with the four different states of the game and how to draw each part. This reduces game1s requirement to know where the game is at. The

SoundManager deals sound including loading and playing the sounds as requested. The keyboard controller was the most coupled class as it deals with many different aspects and thus needs to access many things. To fix the coupling issues and fix the readability of the code a command pattern was set up to deal with the actual logic. The required creating a command interface called ICommand and then adding the commands to a list with the certain key mappings. This pattern eliminated most of the coupling in both controller classes and reduced the duplication of code as the controllers now share the same commands.

Code Metrics

- Week 1: 253 Stylistic Recomendations
- Week 2: 30 Stylistic Recomendations

IDE0045: Convert to conditional expression
dotnet_style_prefer_conditional_expression_over_assignment = false
- Suppressed because doing math in if statement reduces complexity

IDE0044: Add readonly modifier
dotnet_style_readonly_field = false
- Supressed because unknown whether need to be able to modify in the future

IDE0090: Use 'new(...)'
dotnet_diagnostic.IDE0090.severity = none
- Supressed because breaks code if not intialized correctly

IDE0011: Add braces
dotnet_diagnostic.IDE0011.severity = none
- Suppressed because doing math in if statement reduces complexity and clean code

IDE0058: Expression value is never used
dotnet_diagnostic.IDE0058.severity = none
- Suppressed because not usable yet

IDE0028: Simplify collection initialization
dotnet_diagnostic.IDE0028.severity = none
- Supressed because breaks code if not intialized correctly

IDE0055: Fix formatting
dotnet_diagnostic.IDE0055.severity = none
- Suppressed because doing math in if statement reduces complexity

IDE0052: Remove unread private members
dotnet_diagnostic.IDE0052.severity = none
- Supressed because unknown whether need to be able to modify in the future

IDE0032: Use auto property

dotnet_diagnostic.IDE0032.severity = none
- Auto proprerty does not set them up correctly and this needs to be suppresed

IDE0078: Use pattern matching
dotnet_diagnostic.IDE0078.severity = none
- Pattern Matching is not allowed in the current version of C# we are using

IDE0063: Use simple 'using' statement
csharp_prefer_simple_using_statement = false
- This is part is generated monogame and thus we can not modify it

IDE0008: Use explicit type
dotnet_diagnostic.IDE0008.severity = none
- Can not do this because this is part of the monogame

IDE0038: Use pattern matching
csharp_style_pattern_matching_over_is_with_cast_check = false
- Pattern matching is not allowed for current version C#

The 30 or so check styles remaining have no way to suppress their flags. This is a bug noted in my microsoft and there is no fix as of yet. 25 of these are the style for not having simplied if statements which means not doing any math in them or accessing a array. If we were to simplify the ifs the code could lose readability as there would be many extraneous lines of code just store a value to be used once.