Every class specific file is in Enemy folder

Alan:
- Aquamentus.cs
- Boomerang.cs
- EnemySprite.cs
- EnemySpriteFactory.cs
- Fireball.cs
- Goriya.cs
- IEnemy.cs
- Stalfos.cs

Kristin:
- BladeTrap.cs
- Gel.cs
- Keese.cs
- Wallmaster.cs
- ICommand: Lines 218-291 and Enemy reset
- Main Game file (We both worked on it)

Review

Author: Simon Kirksey
Date: Sunday, February 21
Sprint 2

Author: Alan Wu

IEnemy.cs
______Very straightforward interface. Extracting common features like direction, location, and speed to the interface could decrease work from refactoring in future sprints.

Stalfos.cs
______Passing in magic numbers as parameters of the constructor could decrease work needed in refactoring for future Sprints. Using an enum for the direction would increase readability. (one is present in LoZHelpers.cs).

Author: Kristin Bradshaw

BladeTrap.cs, Gel.cs, Keese.cs, Wallmaster.cs
Same comments as Stalfos.cs. Pass magic numbers in constructor and use an enum for direction.

ICommand.cs
SetSpriteEnemy.Execute()

It seems like this command is trying to accomplish multiple goals. I'd split it into a nextEnemy and a previousEnemy command, as ICommands shouldn't be accessing the Keyboard state (since we go through the effort of registering commands with the keyboardController).

Professor Boggus also advised having the creation of enemies be tied to the main game class to keep the ICommands simpler.

For C#, you don't need the { }'s on case statements nor the LegendOfZeldaClone before the enemy constructors.

Recap

Hypothetical Change

Implementing collision detection between IPlayer and IEnemies. Currently, all the enemies have private location values, but extracting these into a property required by the IEnemy interface would make detecting collisions much easier. On a similar note, if enemies can have different sizes (and thus hitboxes), abstracting access to those values to the IEnemy interface would also be advantageous.

Summary

Very concise and readable code. A few recommendations involving the IEnemy interface to give some future proofing for functionality in later Sprints.

Questions

Is there a particular reason you made a sub-namespace for enemies? I don't have any issues with it, just curious.

Time Spent
About half an hour

Code Review author: KT Goldstein
Date: 2/21/2021
Sprint #2
Time to complete review: ~90 minutes

Aquamentus.cs
Author: Alan
Comments on code quality: Simple logic overall, looks good.
Comments on code readability: Good variable naming, easy to read.
Hypothetical change: It might be helpful to have a List<IEnemyProjectile> rather than List<Fireball> for the purpose of maintainability in case you wanted to change what kind of projectile Aquamentus uses later on (I mention this idea of an IEnemyProjectile interface and explain it a bit more later on in this code review). Though this might not be necessary and is more of a design suggestion, since we likely won't have to change the type of projectile.

Boomerang.cs
Author: Alan
Comments on code quality: Simple and understandable logic in Update(), and Draw() relies on ISprite which keeps that method simple as well; for future code expansion the concrete private Goriya might be too specific (as opposed to having a more abstract general IEnemy)
Comments on code readability: Simple and clear naming of variables, looks good.
Hypothetical change: Might want to make Boomerang implement an IEnemyProjectile inte rface rather than IEnemy, since there is another projectile in this sprint as well (Fireball.cs). Additionally, it may help for future project expansion if the private Goriya was instead a private IEnemy (would take some refactoring). A way to implement this (just a thought) could maybe be for line 43 to be **`enemy.Update();`** rather than **`goriya.catchBoomerang();`** (also just a side note that I think by C# naming conventions catchBoomerang() and getGoriyaLocation should begin with a capital letter since they are public methods if you end up keeping these methods). Then if you still use getGoriyaLocation() and/or catchBoomerang() only Goriya would know about these Goriya-specific concrete methods.

EnemySprite.cs
Author: Alan
Comments on code quality: Nice use of C# default values for animationSpeed (line 20).
Comments on code readability: I'm a little confused about what **`private int count`** is used for, perhaps a more descriptive variable name could help, e.g. frameCount or something that describes what it is that it's counting.

Hypothetical change: More descriptive name for variable **`count`** (simple automated refactor).

<u>EnemySpriteFactory.cs</u>
Author: Alan
<u>Comments on code quality</u>: There are a fair amount of private Texture2Ds, and from a code maintainability standpoint it might be better to group them all into one declaration with commas (I think I'm mainly confused about boomerangSprite and stalfosSprite being on the same line). Alternatively, you could pass in a single texture 2D with all of the sprites on it (might have to create your own sprite sheet/texture atlas) so that you don't need so many specific private variables, though if you do this it might be harder to add more enemies later on using just that one same sprite sheet if we wanted to do that.
<u>Comments on code readability</u>: Good variable names, clear and readable.
Hypothetical change: Maybe put all of the private Texture 2D declarations on one line (unless you feel that the slight cost to maintainability is worth it for the readability it provides specifically to your private variables), or create a single sprite sheet that has all of the enemies on it (would probably be a pretty big pain to implement this change at this point though, but it'd get rid of your need for all of those other private Texture2D variables).

<u>Fireball.cs</u>
Author: Alan
<u>Comments on code quality</u>: Simple and concise logic, only thing is the same as I said about Boomerang.cs (implementing an interface other than IEnemy, e.g. IEnemyProjectile).
<u>Comments on code readability</u>: Variable names all make sense, looks good.
<u>Hypothetical change</u>: Same as Boomerang.cs, creating an interface for enemy projectiles and having Fireball implement that interface instead of the IEnemy interface.

<u>Goriya.cs</u>
Author: Alan
<u>Comments on code quality</u>: Similarly as I said about Boomerang.cs, having the private Boomerang be an interface rather than a concrete type might be good in case we wanted to change the projectile.
<u>Comments on code readability</u>: Variable names look good, easy to read with simple logic.
<u>Hypothetical change</u>: Could possibly change lines 46-71 to a switch statement to make it more readable, though that's arguably a matter of personal preference/opinion of readability.

IEnemy.cs
Author: Alan
Comments on code quality: Simple and straightforward
Comments on code readability: Good that the method names are the same and take the same parameters as the typical Update() and Draw() methods; helps with higher cohesion
Hypothetical change: It's possible that it could be helpful to have a public property for an IEnemyProjectile (as mentioned previously regarding Boomerang.cs) that would just have some default value if the enemy doesn't use a projectile, though I'm not certain whether that would be useful.

LegendOfZeldaDungeon.cs
Author: Alan & Kristin
Comments on code quality: Simple logic, nothing extra, easy to follow. There's only a single command for both keys P and O, do you guys need to implement different commands for next/previous enemy sprites?
Comments on code readability: Simple and readable, good variable names.
Hypothetical change: No suggestion, maybe adding a separate command for previous/next enemy sprites if that's part of the Sprint 2 requirements (no reason for it otherwise since we won't use it again).

Overall
Code is readable and of good quality, with simple logic. Main thing I'd suggest is the creation of an IEnemyProjectile interface, and all the refactoring that would come with that (including inserting the interface where concrete types Fireball and Boomerang currently are).