Framework

- Genotype
    - The Genotype has be represented with a list of types
    - The Genotype would contain Genes
        - Genes are created by the specific applications
- GeneBuilder
    - Builds Genes Randomly, must have info to Priority and The Genetic Property to build the Genes
- GeneticProprety
    - Used to hold the Report
    - Used to give Phenotypes a Genitic Property
        - allows the Phenotype to do something or give it a feature
    - Must be adapted to all Genetically Built Objects
    - Allows a Priority to be added to it
    - Returns the Report
- GenerationBuilder
    - takes the GenotypeBuilder and Builds Genotypes from them for the first Generation.
    - Used in the First Generation
    - Given the properties of the GenerationCritieria it completes this
- GenerationCriteria
    - Tells the GenerationBuilder what properties the users is looking to start the Generation with
- Expression
    - Has to create the Phenotype from the list in the Genotype
    - We need to account for Priority or a weighted system for the Phenotypes characteristics
- Priority
    - Used with the Expression to create Phenotypes
    - Addes Priority to the Genetic Object for Weighted Use in the Engine
    - Must be a Way to adapt the Genetic Property of the Phenotype with the Property Simply add it to the Property
        - This allows a Property to have a Priority
- Priority Factory
    - Takes in the part of the Gene that gives a Genetic Property a Priority
- Phenotype
    - uses the Application Specific Objects to do things
- Controls of the Interface (Tasks????)
    - These are the Controls that the Phenotype will use to interact with the Application
- Fitness Test

    - The Fitness Test will test each Phenotype according to Criteria
    - There will be a Leveled system, so the value of “fittest” will be determined by the Difficulty level
    - Phenotypes will be given ReportCards
    - The Report Cards will be evaluated, accordingly
- Report
    - Used to give Statistics to the Phenotype
    - Used by the FitnessTest to evaluate the Phenotypes
    - There is a Report for all Properties
- Evaluation
    - This is the element of the Fitness test that says whether a Phenotype passes or not
- FitnessCriteria
    - Used to give each PropertyReport of the Phenotype a Score
    - Once the Scores are calculated They are used in the Evalutation
- Level
    - a Strategy that calculates the Levels fitness requirements
    - All Phenotypes must pass this Level to go on to the next Generation
    - Used in the Engine to Test Phenotypes
- Criteria
    - Used in the Fitness test
    - A Criteria is an Object that tests the Ability of the Phenotypes in the Fitness Test
        - It does this by testing Properties of the Phenotype
- Mutations
    - Mutate strings according to the Gene characteristics
    - creates a new Genotype
- Mating
    - Must be able to support link Mating, with random mean listing
    - Creates a new Genotype
- Engine
    - This is the Application part
    - It creates the Users and Tests them and produces the results of the Evolution
    - It uses Expressions
    - It uses the GenerationBuilder
    - It Test the Phenotypes with a FitnessTest

Application 1: Player AI for Strategy Game (like civilization, etc)

- Character or Phenotypes;.
    - The user links a character or role (medium that a player plays the game) to the app
    - each Objective given to the Phenotype is given a “priority rating” by the Framework Properties
    - A Player has a list of Strategies that chooses the Properties (Objectives) that are available and then uses one of those Objectives to commit a Task
        - The Objective that has the Highest Priority that we are able to play, and that has not been completed yet
- PriorityStrategy
    - Chooses the Prioritized Objective that has the Highest Priority and is able to be completed in the state of the game, if it is not already completed.
- RulesOfTheGame
    - The app needs Rules of the Game to categorize what is important. Example Money, Diplomacy, Shooting an enemy, shooting friendlies….
    - The Rules of the Game are classes which define Broad Objectives. Example “Pieces to Take” would classify the pieces that can be taken from the opponent. while “Safety” would classify the amount of risk you are willing to take to get that piece.
    - Rules of the game can have State Rules for particular states of the game, phsases ….
- Genotypes
    - Uses Genes
        - Genes are created from the selected Objectives
    - Genotypes use a list of strings to represent the Genotypes
    - The Genotypes then go through the Expression phase to create the Phenotype.
- Gene
    - A gene is a representation of a Objective for a Genotype
- A GeneFactory
    - Creates the Genes for the Genotype
    - Randomly creates Genes based off the Rules of the Game
- A GenotypeBuilder
    - Given a Randomizer it creates the Genotypes
        - Randomly create a Objective for the Gene
        - Uses the GeneFactory to randomly create these Genotype
- Priority System
    - This is determined by the Genotype string to build the Phenotype
    - The Priority is added to each Objective within the Phenotype

    - There is a strategy to decide which Objective has the greatest Priority, based on whether the Objective can occur, if it can't, that Objective cannot occur
- Objectives
    - Now each Objective is created by the Rules of the Game that allows the game to manipulate the Objective according to the genotypes used. Example: in Chess Objectives: 1. Take opponent's bishop within 3 moves of safety. 2. Take queen as soon as possible. created from the Rules "Pieces to Take" "Safety".
    - A Objective can be generated from the Rules of the Game in the following fashion. Build Rule with other Rules applied ----> Objective. This allows the Objectives to get really complex yet be based off of simple ideas.
    - Now we have to note that a Objective can be made of multiple Rules of The Game, and to make it work all Rules of the Game must apply for the Objective to be complete. Because of this the Player will decide on making a Task and then make more if applicable to the other Rules. if all Rules are completed then it commits the Tasks, this is a Transaction. Example: Make checkmate within 2 moves but don't allow sacrifice.
    - Another thing to note is that some of the Rules of the Game are of the same category or contras to each other. Example: "Don't allow sacrifice" and "Implement a Bait"
    - These contras, cannot contradict each other. So under the respective category of Rules of Game, they would implement the same sub class under the Rules of Games interface.
        - To deal with contras, we can test the Objectives in the Builder, if it complete the building it will send a true value stating it was built, else a false value
    - Objective or something needs to contain the Reports used by the Fitness Test
- Expressions
    - The Expressions take the Genotype (list of String Objectives) and create the Player.
    - The Expressions take each Objective of the Genotype and create the Player,
    - each Objective given to the Phenotype is given a priority rating.
        - The reason we create the priority rating system is that we need to weigh all objectives within the game, it is very important that there is no apsense of knowledge within the game, because one perspective can be different from another.
    - The Expression creates the Computer controlled Player, the way to interact with each Objective need to be different corresponding to each Rule of the Game created.
- Tasks
    - The game is played using the User interface, which is linked to the Objective, which is linked back to the Rules of the Game. The Rules of the Game contains the information on how to accomplish the Objective. Example:

Contains the knowledge to click to buy power ups….. , to move to the Objective……
    - Therefor the Rules Of the Game needs a list of Tasks, which run the commands to get the player closer to the Objective.
    - Now the Player will commit a Objective and the Task will complete a step closer to the objective, what happens if the Objective becomes void, it will go after another. What happens when there are multiple steps to take of per "Turn" or unit of time?? Example: Mario and move forward and fire a fireBall at the same time.
        - So a Player can commit multiple Objectives as long as the game completes them, then we move closer to the Objective perspective to the game.
    - The Objectives commit Tasks as long as they are able to. Now we have to take in account for the opponents to take turns or move as well.
    - Tasks can have State of the Game associated to them as well. To enable certain Tasks during certain stages of the game
    - In order for the Objectives to take turn they go to the Rules of the Game, which will compute the necessary information to conduct the next Task for the following Objective. This will more or less ask as a Chain of commands. Example: Objective make opponent get into checkmate. Move nearest piece to put into checkmate ----> move a piece for next move to be checkmate -----> move nearest piece to be within two moves of checkmate…….
- Game Generation
    - The start of the series of games that are played
- Generation Criteria
    - The Criteria that defines the state of the application, and how many levels it will finish and such...
- Mutation
    - Can add to the Objective String List of a Genotype
    - can remove
    - can mutate the objective strings
        - Mutation would require that the Gene is change
- Mating
    - Mating will add the mean ratio of the two phenotypes Genotypes String Objectives, it will take those at random.
- Fitness Test
    - The fitness test needs to test several things to determine the best candidate of the game.
        - This can be by counting the number of Objectives completed
        - Winning the actual game
        - Other statistics of the game can be included
    - WaystoWin is the Criteria
        - # of Objectives completed

        - Weight of Objectives
        - Time ……
- Level
    - changes the requirements for the Level
    - Used by the Fitness test to pass all the users
    - Takes in all the Players and evaluates whether the level will increase or if the players are not passing the previous level evaluations
- Reports
    - Need to evaluate the Objectives, and keep track of all Tasks operated under the Objective
    - The report is inside the Objective
- Game Engine
    - Contains all the information for the game to go on
    - Provides the First generation, to create the First set of Genotypes
    - Contains the Entire set of Rules which is needed to create Genes
    - Contains the Mating Sequence
    - Contains the Mutation Sequence
    - Contains the Fitness Testing
    - Contains the List of all Players
    - Contains the List of all Players that pass a Level
    - Contains the Player Expression Factory

Application 2: User Interface Analyzer

- Application information
    - The purpose of the application is to analyze how good a user interface is, and evaluate potential improvements.
    - The program works as a plugin for the server where the website is hosted.
- How It Works
    - Different phenotypes (web pages) are presented to users of the website
    - Data will be taken from user interaction over a certain period of time. This data includes how the webpage is used, as well as a brief survey on overall user experience.
    - The user input is analyzed at the end of each generation, and a fitness level is determined for that specific phenotype
    - Phenotypes are mated and mutated based on fitness level, and new phenotypes are born
    - repeat
- Genotype
    - A genotype is composed of multiple genes, which at it's most basic level is composed of multiple css properties.

        - a single gene might be represented with the properties
            - text-align: left
            - margin: 0 auto
            - background-color: gray
    - Genes are represented as the different HTML tags
        - Modifiers like classes and id's can be applied to the HTML tags
        - Each of the CSS properties associated with that class are now apart of the HTML tags genetic makeup
- Phenotype
    - The phenotype is the physical representation of Genotype
    - In our case, this means a webpage
        - Although the physical representation is what the website looks like, on the application level the phenotype is treated as the html and css file itself
    - Important Note: Each "Phenotype" is a specific webpage on the website. Different Phenotypes of the same species are different versions of the same webpage (accessed from the same URL). When discussing this application, only phenotypes (or genotypes) of a single species (webpage url) interact with eachother.
- Changes/Allowable Modifications
    - The changes that can be made are applied to the DNA (CSS Properties)
        - Mutations occur in the DNA of the genotype, and are applied to the phenotype in the css file
    - Some Structural Properties can only be modified
        - width
        - height
    - All other properties can be modified, added, or removed (All other CSS Properties)
        - text-color
        - background
        - etc.
- Mutations
    - Mutations randomly occur at the end of each generation
    - Mutations are limited to the types of allowable modifications
- Mating
    - Mating occurs after each generation and is based on fitness level
    - The fitness level is used to determine which genotypes mate
    - changes here are also restricted to the allowable modifications
- Fitness
    - Fitness will be determined from analysis of user interaction as well as a survey
        - Data collected from user interaction
            - access time
            - frequency of access

        - Feedback via some a survey
            - On a scale of 1 - 100 how visually pleasing was this webpage?
            - On a scale of 1 - 100 how easy to navigate was this webpage?
            - On a scale of 1 - 100 how was your overall experience navigating the webpage?
- Phenotype to Genotype Conversions
    - HTML and CSS Parsers exist to convert the phenotype into it's genotype representation
    - When Parsing HTML
        - First add the ID's, and Classes to the composit as decorators
        - Then add the html tag as the Gene itself
    - CSS
        - parse the tags/ classes/ ID's
        - Look at the Composite for the genes that fit the tag.
            - for each genes, append the styles.
- Genotype to Phenotype Conversion
    - HTML and CSS file is created from all the tags in the composite structure