AutoBattle - Documentation

- Remoção de variáveis locais na classe Program (*playerCharacterClass*, *EnemyCurrentLocation*, *PlayerCurrentLocation*, *PlayerCharacter*, *EnemyCharacter*, *numberOfPossibleTiles*)
- Enums movidos para classes separadas

- Criação de um Helper para funções genéricas
- Realocação das funções **Program.GetRandomInt()** e **Program.GetPlayerChoice()** para a classe **Helper.cs**
- Refatoração da função **Helper.GetRandomInt()** para evitar a criação do objeto Random cada vez que for chamada
- Refatoração da função **Helper.GetPlayerChoice()** para validar a escolha dentre os elementos disponíveis no enum CharacterClass. Facilitando assim a criação de novas classes.

- Alteração na função **Character.TakeDamage(float)** para receber o dano devido
- Criação de constructor para a classe **Character.cs**
- Conversão de variáveis para properties na classe **Character.cs**
- Criação de variáveis e properties adicionais para a classe **Character.cs** (*CurrentBox, idDead, Simbol, Team, Class, _closestTarget, _targets*)
    - CurrentBox salva as informações a respeito da posição atual do character;
    - idDead verifica se a variável Health é <= 0;
    - Simbol é utilizado para desenhar o ícone do character no grid;
    - Team para saber a qual time o jogador pertence;
    - Class para saber a qual classe o jogador pertence;
    - _closestTarget é preenchido ao iniciar um movimento, é usado tanto para saber pra onde o character deve ir ou se está apto a atacar;
    - _targets é uma lista de Characters que pertencem ao time inimigo
- Refatoração do método **Character.Attack()** para levar em conta o damageMultiplier
- Criação do Método **Character.HandleClassChoice()**, para alterar os status dependendo da classe
- Conversão do método **Character.StartTurn()** para **Character.DoAction()**, utilizado para definir qual será a ação do character naquele turno.
- Criado método **Character.ShowStats()** para mostrar os status do character ao final de cada turno.
- Refatoração do método usado para checar targets próximos. Agora ele compara a distância entre a própria coordenada e as coordenadas dos targets para verificar qual se encontra mais próximo.
- Criação do método **Character.MoveTo()**, utilizando coordenadas(int, int) como parâmetro.
- Refatoração do sistema para localizar a próxima casa pra se locomover, ao invés de usar uma lista com index como base, a variável **Grid.grids** foi transformada em **Dictionary<(int, int), GridBox>** onde a tupla (int,int) é a chave de acesso para a respectiva gridBox. Onde a tupla representa as coordenadas do grid

- Separação de classes no arquivo **Types.cs**

- Remoção das variáveis de index da classe **GridBox.cs** (Substituido pela tupla(int,int) *Coordinates*)
- Criação das funções **GridBox.SetOcupied()** e **GridBox.GetDistanceToOtherBox()** como métodos auxiliares usados por outras classes.

- Alteração do método **Grid.DrawBattleField()** para incluir o símbolo de quem está ocupando a casa
- Adicionado dois métodos auxiliares **Grid.Exists()** e **Grid.SetGridOcupation()**, ambos usam coordenadas (int, int) como parametro, e servem para checar se determinada coordenada é válida e alterar o estado de ocupação da mesma.

- Na classe **Program.cs** remoção dos métodos **AlocateEnemyCharacter()**, **AlocatePlayerCharacter()**, **CreateEnemyCharacter()** e **CreatePlayerCharacter()**. Todos foram reduzidos ao método **CreatePlayers()**
- Adicionada cláusula para checar o fim de jogo no loop de update dos players

- Criação da classe **TeamManager.cs** para lidar com relacionados a feature extra de *”Add more characters and divide them into different teams.”*
- A classe é responsável por Separar os players em times pré determinados, e setar preencher a lista de targets em cada character com todos os players dos times adverários (é possível mais de dois times coexistirem no tabuleiro).
- Responsável também por checar quando o jogo é finalizado, sempre que um jogador morre ele é removido do time, se o número de pessoas no time é zero então o time inteiro é elimidado, ao sobrar apenas 1 time o jogo é encerrado.
- Responsável também por checar possível vitória por WO, onde há apenas 1 time no início do jogo.