Tests

**GridModelTest**

* GenerateGrid
  + Check generated at row 1 & 9
  + Check generated at column 1 & 9
  + Check generated at mixed row and col
* numberOfDuplicate
  + Check duplicate in just row
  + Check duplicate in just column
  + Check duplicate in just subgrid
  + Check duplicate in mixed condition
  + Check there is no duplicate
* gameFinished
  + Check game matching at one subgrid
  + Check game matching one row
  + Check game matching one column

**GridControllerTest**

* CellControllersInit()
  + Check the this.\_cellControllers[r, c].Model == \_gridModel.Cells[r, c]
* NumberControllersInit()
  + Check int 1 to 9 from the this.\_numberControllers.number
* BuildGrid()
* FillNumber()
  + Fill number when timer is paused
  + Fill number when no cell is selected
  + Fill number of unchangeable cell
  + Fill random number
  + Fill the last digit of a specific number (like 9)
* UndoLastAction()
  + Simulate fill in number and undo them to check
* ResetGrid()
  + Check the gridModel.Cells still match all this.\_cellControllers[r, c].Model
* UpdateNumberBarVisibility()
* UpdateNumberColor()

Classes

**CellController**

+ currentlySelected: CellController {static}

+ editorRow: int

+ editorCol: int

- \_cellModel: CellModel

- \_isUnchangable: bool

- \_numberPrefab: GameObject

+ CellModel get set

+ IsUnchangable get set

+ CellController(): Constructor

- Awake(): void

+ SelectThisCell(): void

+ FillNumber(color:string, init: bool): void

+ HighlightCell(color: string): void

+ RemoveHighlightCell(): void

- InstantiateNumberMaterial(color: string): void

+ UpdateModel(num: int): CellController

**CellModel**

- \_num: int

- \_sol:int

- \_row: int

- \_col: int

+ CellModel(num:int, sol:int, row:int, col:int)

+ Public get set methods

**GameLog**

- \_instance: GameLog {static}

- \_sw: StreamWriter

- \_dirPath: string

- GameLog(): Constructor

+ Instance get {static}

- Initialise(): void

+ WriteToLog(msg: string): void

+ CloseLogger(): void

**GameManager**

- \_gridController: GridController

- \_timerController: TimerController

- \_leaderboardController: LeaderboardController

- \_hasGameCompleted: bool

+ Instance: GameManager {static}

- Awake(): void

- Start(): void

- Update(): void

- OnApplicationQuit(): void

+ GridController get

+ Timer get

+ IsTimerPaused(): bool

+ RestartGame(): void

**GridController**

- \_gridModel: GridModel

- \_cellControllers: CellController[,]

- \_numberControllers: List<NumberController>

- Start(): void

- Update(): void

+ Init(): void

- CellControllersInit(): void

- NumberControllersInit(): void

- BuildGrid(): void

+ FillNumber(number: int): void

+ UndoLastAction(): void

+ ResetGrid(): void

+ PushUndoState(controller: CellController): void

- NumberColor(duplicateExist: bool): string

+ UpdateNumbberBarVisibility(): void

+ IsGameFinished(): bool

+ IsNumberFullyUsed(num: int): bool

**GridModel**

- \_cells: CellModel[,]

- \_puz: int[]

- \_sol: int[]

- \_actionStack: Stack<UndoAction>

- \_numCount: int[]

+ GridModel(): constructor

+ Cells: CellModel[,]

+ Puz: int[]

+ Sol: int[]

+ Init(): void

+ SelectPuzzle(filePath: string, puzId: int = -1): GridModel

+ GenerateGrid(puz: int[] = null, sol: int[] = null, isReset: bool = false): GridModel

+ DuplicateExists(num: int, row: int, col: int): bool

+ IsGameFinished(): bool

+ TryPopLastAction(out action: UndoAction): bool

+ PushLastAction(num: int, row: int, col: int, color: string): void

+ CalculateDigitUsage(): void

+ IsNumberFullyUsed(num: int): bool

+ ResetGrid(): GridModel

**MenuButtonController**

- \_gameManager: GameManager

- \_pauseResumeLabel: TMP\_Text

- \_pauseResumeIcon: FontIconSelector

+ Start(): void

+ Update(): void

+ OnUndoButtonPressed(): void

+ OnPauseButtonPressed(): void

+ OnRestartButtonPressed(): void

**NumberController**

+ number: int

- \_gridController: GridController

+ Initialize(grid: GridController): void

+ OnNumberPressed(): void

+ SetNumberGameObjectVisibility(visibility: bool): void

**NumberDatabase**

+ Instance: NumberDatabase {static, get; private set;}

- \_numberDict: Dictionary<int, GameObject>

+ entries: List<NumberEntry>

- Awake(): void

- BuildDictionary(): void

+ GetNumber(num: int): GameObject

class NumberEntry

+ num: int

+ prefab: GameObject

**PuzzleReader**

- \_puzzle: List<int[]>

- \_solution: List<int[]>

+ Puzzle: List<int[]> { get; set; }

+ Solution: List<int[]> { get; set; }

+ PuzzleReader(): constructor

+ ReadCSV(filePath: string, numPuz: int = 100): void

**SoundEffectDatabase**

+ Instance: SoundEffectDatabase {static, get; private set;}

+ sfxSource: AudioSource

- \_sfxDict: Dictionary<int, AudioClip>

+ entries: List<SoundEffect>

- Awake(): void

- BuildDictionary(): void

+ GetAudio(id: int): AudioClip

+ PlayAudio(id: int): void

class SoundEffect

+ id: int

+ name: string

+ audio: AudioClip

**TimerContainerController**

- \_pos: int

- \_timerContainer: GameObject

+ TimerContainer: GameObject { get; set; }

+ Position: int { get; }

+ Start(): void

+ Update(): void

+ DisplayDigit(digits: int): void

**TimerController**

- \_timerModel: TimerModel

- \_timerList: TimerContainerController[]

+ Model: TimerModel { get; }

+ Start(): void

+ Update(): void

+ Init(): void

+ DisplayElapsedTime(): void

+ PauseGame(): void

+ ContinueGame(): void

+ IsPaused(): bool

+ RestartTimer(): void

- ConvertTimeStringToIntArray(time: string): int[]

- BuildTimerNumberControllerList(): void

**TimerModel**

- \_timer: Stopwatch

- \_isPaused: bool

+ TimerModel()

+ IsPaused: bool { get; set; }

+ GetElapsedTime(): string

+ GetElapsedTimeFloat(): float

+ PauseTimer(): void

+ ContinueTimer(): void

+ RestartTimer(): void

**TimerNumberDatabase**

+ Instance: TimerNumberDatabase {static}

+ entries: List<NumberEntry>

- \_numberDict: Dictionary<int, GameObject>

- Awake(): void

- BuildDictionary(): void

+ GetTimerNumber(num:int): GameObject

class NumberEntry

+ num: int

+ prefab: GameObject