

引擎工作循环：

idle状态，等待tile.button触发Select操作。

Select操作：仅当不处于Swapping, Matching,Shuffling状态时：

当\_selection填满两个Tiles后，SwapAsync(tile1,tile2)

若交换后无match，SwapAsync(tile1,tile2)

当无可用操作或有match时：

Shuffle生成新tile：进入Shuffling状态

SwapAsync(tile1, tile2)

TryMatchAsync()：进入isMatching状态。

Do{

FindBestMatch() 返回满足匹配条件的Tiles[] match

基于match播放shrink动画

原地生成新tile并刷新Matrix

基于match播放swell动画

}

While(match不为空) //即重复消除直到无可用match

Update()检查是否进入死局：

以isMatching、isShuffling、isSwapping均为false为前提

FindBestMove()=null→无可行操作 或

FindBestMatch()

若有，回到idle状态

若无，Shuffle重置棋盘并回到idle状态

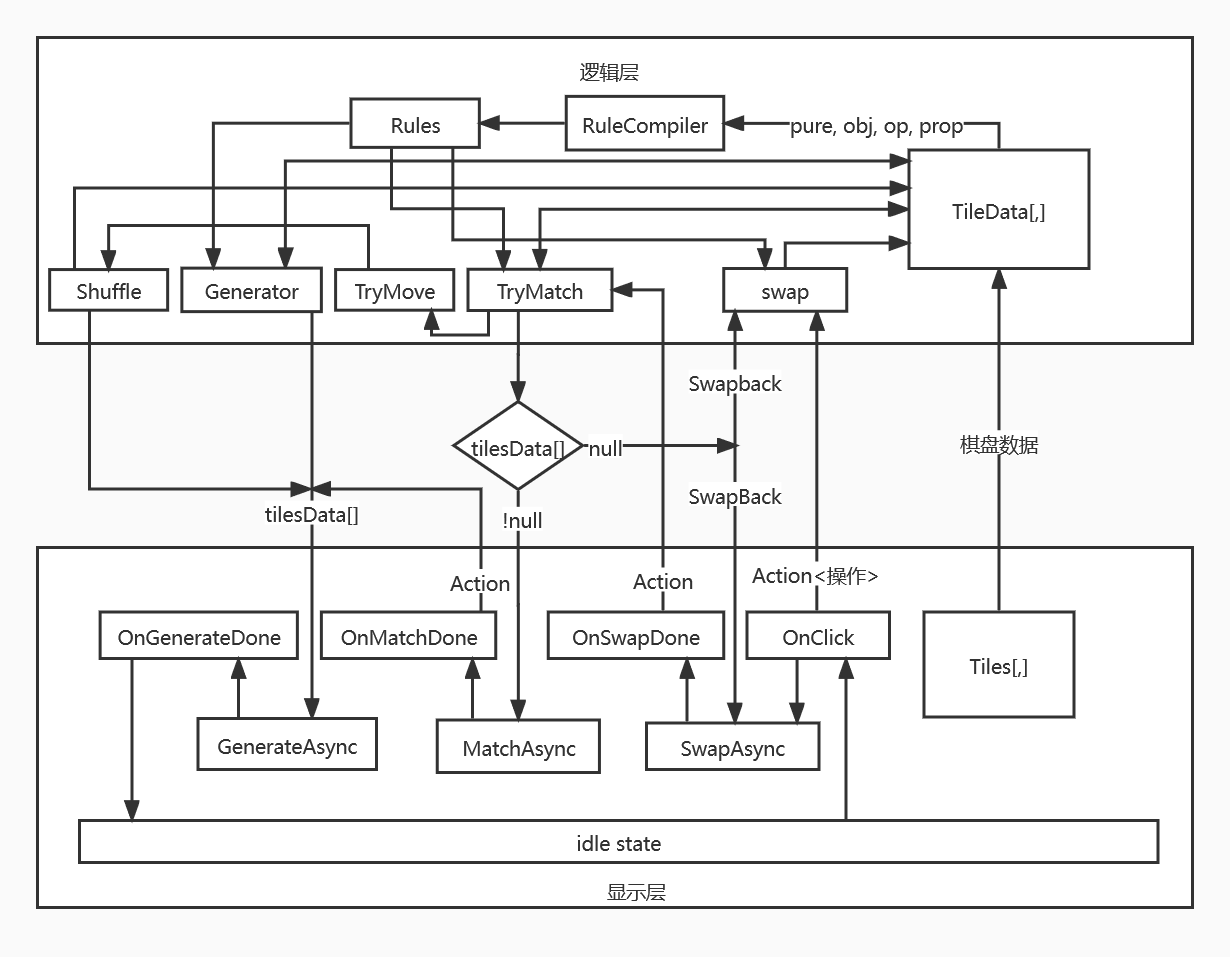
Board数据组织方式

Board→Row[]→Tile[];

引擎如何提取Board数据：GetTile(x,y).data; //返回类型为TileData

引擎如何组织Board数据：TileData[,] Matrix数组

逻辑层（数据）和显示层（棋盘）应当解耦，逻辑层只负责TileData[,]的处理，并生成数据操作，显示层负责将实际棋盘的异步动画显示和数据提取



引擎工作循环：

idle状态，等待tile.button触发Select操作。

Select操作：仅当不处于Swapping, Matching,Shuffling状态时：

当\_selection填满两个Tiles后，SwapAsync(tile1,tile2)

若交换后无match，SwapAsync(tile1,tile2)

当无可用操作或有match时：

Shuffle生成新tile：进入Shuffling状态

SwapAsync(tile1, tile2)

DetectRules() // 遍历Matrix，检测所有完整的语句

CompileRules() // 遍历所有语句，编译规则集并修改现有规则集。

TryMatchAsync()：进入isMatching状态。

Do{

FindBestMatch() 返回满足匹配条件的Tiles[] match

基于match播放shrink动画

原地生成新tile并刷新Matrix

基于match播放swell动画

}

While(match不为空) //即重复消除直到无可用match

Update()检查是否进入死局：

以isMatching、isShuffling、isSwapping均为false为前提

FindBestMove()=null→无可行操作 或

FindBestMatch()

若有，回到idle状态

若无，Shuffle重置棋盘并回到idle状态

规则编译时机：

引擎工作循环：

idle状态，等待tile.button触发Select操作。

Select操作：仅当不处于Swapping, Matching,Shuffling状态时：

当\_selection填满两个Tiles后，SwapAsync(tile1,tile2)

if (!TryMatchAsync()) SwapAsync(tile1,tile2)

当无可用操作或有match时：

Shuffle生成新tile：进入Shuffling状态

SwapAsync(tile1, tile2)

TryMatchAsync()：进入isMatching状态。

Do{

Compiler.Compile()

FindBestMatch() 返回满足匹配条件的Tiles[] match

基于match播放shrink动画

原地生成新tile并刷新Matrix

基于match播放swell动画

}

While(match不为空) //即重复消除直到无可用match

Update()检查是否进入死局：

以isMatching、isShuffling、isSwapping均为false为前提

FindBestMove()=null→无可行操作 或

FindBestMatch()

若有，回到idle状态

若无，Shuffle重置棋盘并回到idle状态