# SDG围棋工程封装库文档

时间：2017.12.13

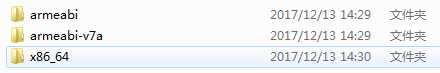
开发者：蒋信厚

Unity版本：2017.1.0f3(64bit)

### 说明

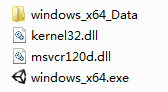
暂时未制作供iOS设备运行的GNUGo库的静态.a库，方法为使用cmake工具编译源码导出xcode工程，利用xcode和iOS的SDK编译.a静态库。可能跟cmake，xcode的sdk等版本有关。

安卓设备需要的.so和windows pc需要的dll库已制作完成，可直接使用或重新编译：



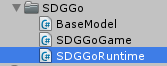
C:\Users\jiangxinhou01\AppData\Roaming\Tencent\Users\919575700\QQ\WinTemp\RichOle\MM3IDB6_UTN~IJ6L_@87W5N.png

其中打windows的PC包后，要补加kernel32.dll和msvcr120d.dll两个系统库，否则在其他pc机上可能缺失依赖库而无法正常运行：



### 接口的封装

最顶层接口封装在了Unity工程SDGGo目录下的三个脚本中：



BaseModel用于定义基本数据结构；

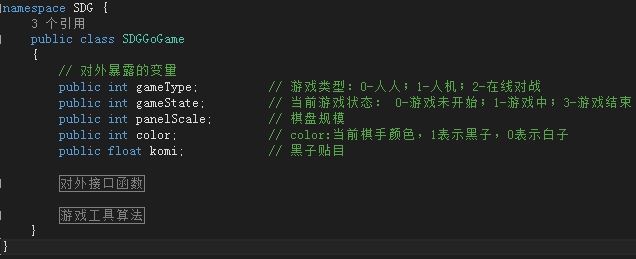
SDGGoRuntime封装了运行时和GNUGo库对接的函数接口；

SDGGoGame定义了后续开发围棋应用的所有需要的接口，可以只关注此文件快速开发新的围棋Demo。

首先基于GNUGo库源码封装的接口函数有以下几个，具体见SDGGoRuntime.cs脚本中的声明：

|  |
| --- |
| // 初始化gnugo,参数为棋盘大小，一般为19  [DllImport("gnuGo-3.8")]  public static extern void SDGGoInit(int boardsize);  // 获取当前打分，正数为黑子领先的目数，负数为白子领先的目数  [DllImport("gnuGo-3.8")]  public static extern float SDGGetScore();  // 在gnugo棋盘上指定位置落子  [DllImport("gnuGo-3.8")]  public static extern int SDGPlayMove(int i, int j, int color);  // 悔棋n步，返回1表示成功，0表示失败  [DllImport("gnuGo-3.8")]  public static extern int SDGUndoMove(int n);  // gnugo落子一步并返回一维落子坐标，如果落子失败返回-1  [DllImport("gnuGo-3.8")]  public static extern int SDGGenComputerMove(int color);  // 判断GNUGo落子合法性，返回1表示可以落子，0表示不可以  [DllImport("gnuGo-3.8")]  public static extern int SDGIsAllowedMove(int i, int j, int color);  // 返回指定位置棋盘棋子状态，0:白子 1:黑子 -1:无子  [DllImport("gnuGo-3.8")]  public static extern int SDGBoardStat(int i, int j); |

Unity端最顶层使用到的接口函数全部封装在SDGGame类脚本中，只需要维护一个SDGGame实例使用其提供的接口函数即可完成所有的游戏逻辑，开发出基本的人-人和人-机围棋对战程序。接口函数和公开变量有如下：



**公开变量：**

|  |
| --- |
| // 对外暴露的变量  public int gameType; // 游戏类型：0-人人；1-人机；2-在线对战  public int gameState; // 当前游戏状态： 0-游戏未开始；1-游戏中；3-游戏结束  public int panelScale; // 棋盘规模  public int color; // color:当前棋手颜色，1表示黑子，0表示白子  public int komi; // 黑子贴目 |

**接口函数：**

核心接口：

|  |
| --- |
| /// <summary>  /// 构造函数，初始化游戏实例  /// </summary>  /// <param name="\_gametype">游戏类型（人-人，人-机，在线对战）</param>  /// <param name="\_scale">棋盘规模（19x19)</param>  public SDGGoGame(int \_gametype, int \_scale);  /// <summary>  /// 获得指定位置的棋子状态  /// </summary>  /// <param name="index">坐标</param>  /// <returns>返回棋子位置状态：黑子、白子、空</returns>  public int GetPanelColor(Point index);  /// <summary>  /// 逻辑棋盘落子操作  /// </summary>  /// <param name="index">落子位置</param>  /// <param name="color">棋子颜色</param>  /// <returns>是否落子成功</returns>  public bool SetMove(Point index, int color);  /// <summary>  /// 悔棋n步  /// </summary>  /// <param name="n">是否悔棋成功</param>  public bool UndoMove(int n);  /// <summary>  /// gnugo智能计算指定颜色最优的落子点落子  /// </summary>  /// <param name="color">AI棋手的颜色</param>  /// <returns>AI想到的落子位置</returns>  public Point GetGenComputerMove(int color);  /// <summary>  /// 获取当前得分  /// </summary>  /// <returns>分值，大于0说明白棋领先，反之黑棋领先</returns>  public float GetCurrentScore() ; |

小工具接口：

|  |
| --- |
| /// <summary>  /// 获取当前玩家对手  /// </summary>  /// <returns>对手颜色值</returns>  public int GetOppenentColor();  /// <summary>  /// 切换当前玩家  /// </summary>  public void ChangeColor();  /// <summary>  /// 坐标是否有效  /// </summary>  /// <param name="p">坐标</param>  /// <returns>是否有效</returns>  public bool IsPointAllowed(Point p);  /// <summary>  /// 坐标溢出矫正  /// </summary>  /// <param name="p">坐标</param>  /// <returns>矫正后的坐标</returns>  public Point PointCorrect(Point p); |

### 库的封装编译和使用

见各目录下的readme文件。