1. Google Blockly

<https://developers.google.com/blockly/>

1. 我们的demo

iKCoder/demos/race-game/index.html

iKCoder/demos/pacman-game-setting/index.html

iKCoder/demos/pacman-game-playerpath/index.html

1. 我们的WorkStation Pages

iKCoder/demos/MainPageDemo\_3/index.html

这里挂的是pacman-game-playerpath这个demo

1. 一些当前我们的基础想法：
   1. Blockly与Scene(即我们提供给使用者介入的游戏) 没有直接关系。
   2. Scene来源于我们对某个CANVAS游戏的包装(壳)和部分适应性修改。
      1. 包装和修改的目的在于抽取出可控制的元素，这其中包括：外观，背景，行为，反应，地图的设定等。
      2. 当前采取的方式比较简单粗暴，都是针对性的修改，没有统一的框架和规范
      3. 有简单的目录结构体系和文件命名惯例，可参看代码。
   3. Blockly和Scene关联建立在且仅在这个壳上，即blockly和游戏本身不发生任何关联。仅通过壳衍生的函数控制游戏元素的变化。

例如：

游戏中，我们添加了一个配置项去控制吃豆人的颜色，playerColor

壳中，我们添加了一个函数 setPlayerColor();

那么，使用者拖动我们定义的一个blockly中的block放入blockly的workspace时他将得到一行代码：

SetPlayerColor(’#ff0000’);

* 1. Blckly的block是可以自定义的，我们通过自定义block来实现从blockly中输出特定代码。
  2. 自定义block可以使用下面的demo来辅助：

http://localhost/google/blockly/demos/blockfactory/index.html

请部署google blockly在本地。(在IIS下建立虚拟目录google指向解压后的google文件夹)

1. 在IIS下建立虚拟目录ikcoder指向iKCoder.WebApplications\iKCoder
2. 关于blockly本身的技术细节：
   1. 其本身的workspace和绘制是基于SVG的。
   2. 对于当前，我觉得不需要过于追究blockly的技术细节，仅熟练掌握自定义block的设计和编码即可
   3. 一般定义一个block仅需要以下俩个部分：
      1. Block本身的细节：这部分可以借助blockfactory来搞

Blockly.Blocks['pacman\_map\_wall'] = {

init: function () {

this.appendDummyInput()

.appendField("Set Wall");

this.setNextStatement(true, null);

this.setTooltip('');

this.setHelpUrl('http://www.example.com/');

}

};

* + 1. Block生成什么代码，这部分衔接到壳上

Blockly.JavaScript['pacman\_map\_wall'] = function (block) {

return 'Scene.initWall();\n';

};