# 课程设计说明

## 1、功能设计

### 1.1 概述

I Wanna Be The Guy 是一款像素风横板跳跃闯关游戏。玩家通过PS/2键盘控制角色移动、跳跃、躲避危险从而到达终点。游戏由学生利用Verilog和SWORD实验板自主开发，利用PS/2键盘进行输入，并通过VGA接口进行输出。

### 1.2 详细描述

1、输入功能：玩家可以使用PS/2键盘来控制角色的移动、跳跃和射击，具体键位设置如下：

* “A”键：向左移动
* “D”键：向右移动
* “W”键：向上跳跃
* 空格键：发射子弹

这一部分将由键盘输入驱动模块实现。

1. 游戏流程：

进入游戏开始界面，按下任意键后，玩家可进入游戏。此时场景和角色会被初始化，开放玩家的控制权限，死亡次数被设置为0。

进入关卡后，角色在陆地上行走并跳跃，目标是抵达终点。碰到终点处的红色按钮即可通关，进入游戏通关界面，同时显示死亡次数。在游戏通关界面，按下任意键即可返回游戏开始界面。

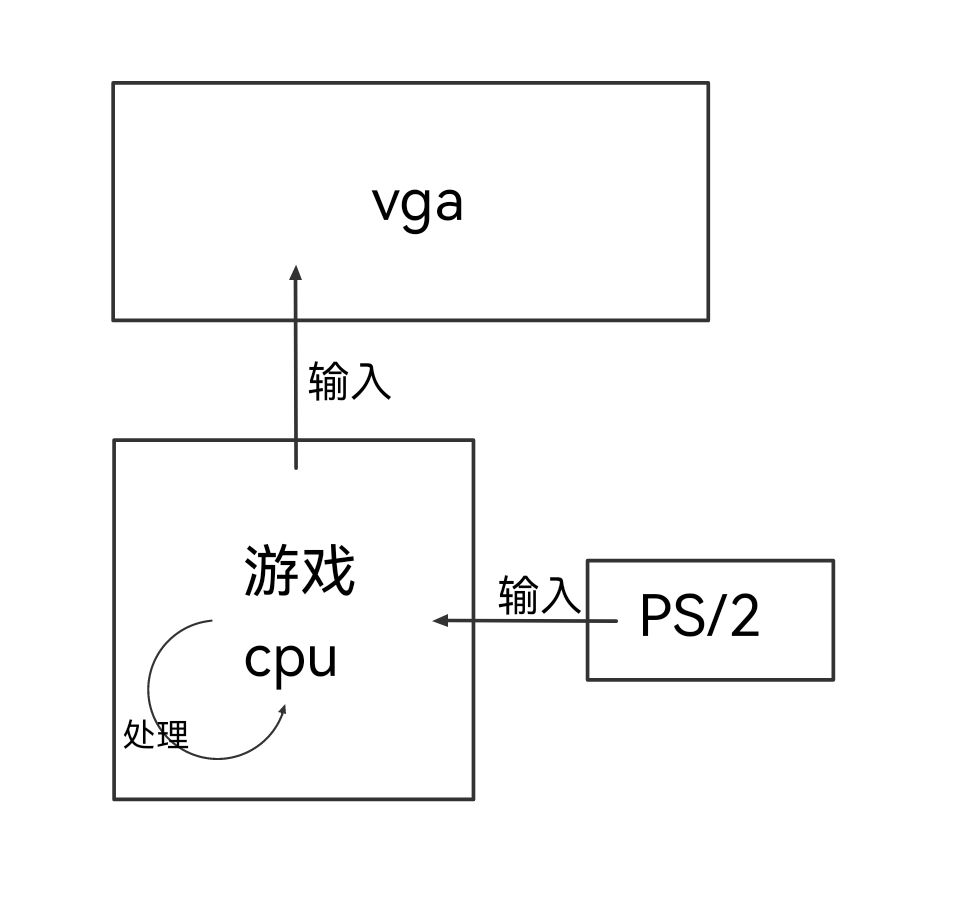
在游戏过程中，角色在途中经过树上的水果的正下方或正上方时，会触发水果向下或向上坠落，角色触碰到水果将会死亡，同时本局游戏结束，死亡计数+1，取消控制权。此时玩家会进入游戏失败界面，同时显示死亡次数，按下任意键即可重新开始游戏。

在任意时刻，按下Esc键，即可返回游戏开始界面。

3、输出结果：通过VGA接口，显示分辨率为640x480、刷新率为60Hz的游戏画面。在每个同步信号的有效数据段内，通过判断当前的游戏状态和扫描位置，决定当前像素的RGB值。

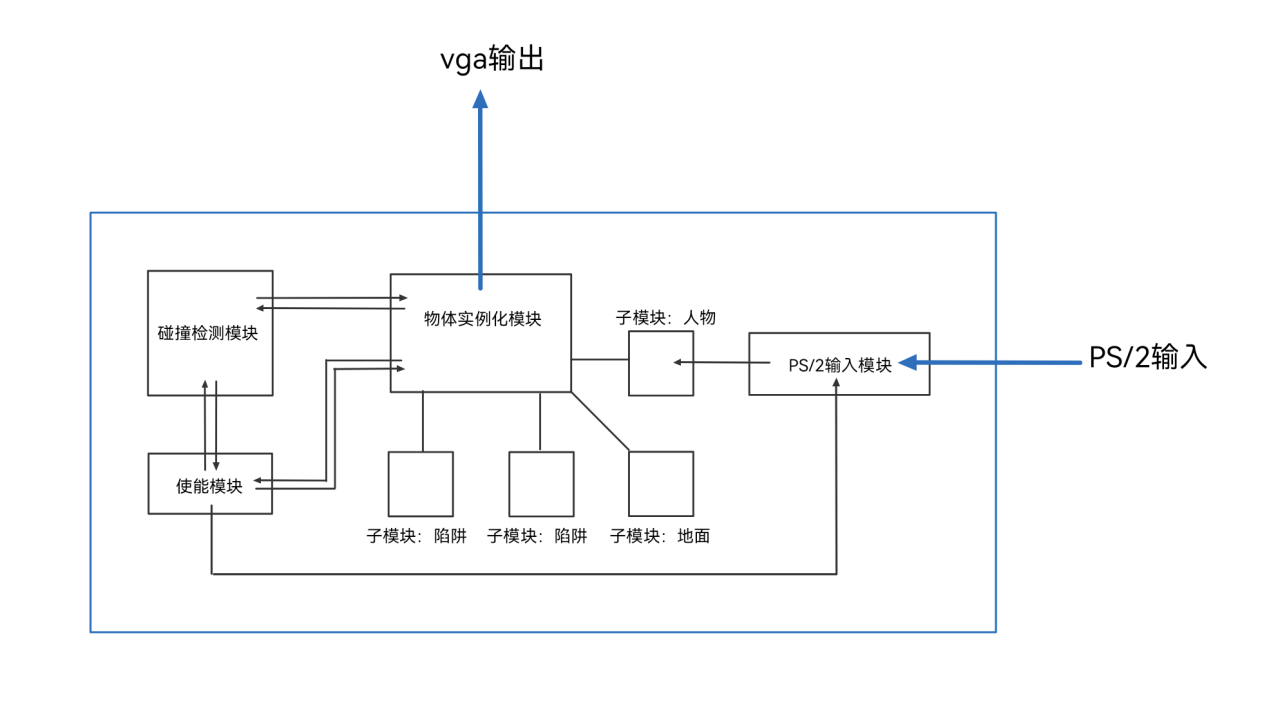
## 2、体系结构设计

### 2.1 硬件体系结构

**

玩家的操作将由PS/2键盘输入到CPU，被游戏程序处理。游戏程序将根据RAM中存储的实时游戏状态，将ROM中存储的图像输出到VGA显示器进行实时反馈。

### 2.2 软件体系结构

**

输入检测模块对键盘输入进行处理，能够改变游戏状态，改变角色的运动方向，创建新的子弹实例。

实例初始化模块负责初始化物体与角色的位置和状态，并初始化整局游戏信息。

碰撞检测模块检测角色与其他实例的碰撞，并由此改变游戏状态和角色运动状态。角色与地面的碰撞将会改变角色的运动；与陷阱的碰撞则会使本局游戏结束。

使能模块通过输入判断游戏进行的状态并输出各个模块的使能信号。角色死亡或通关时使输入模块无效化，而在重新进入关卡后有效化；进入关卡时同时有效化碰撞模块和实例化模块；为了节省算力，可以采用角色靠近陷阱附近仔启用角色/陷阱的碰撞检测，而远离时关闭。

## 3、硬件模块设计

### 3.1 PS/2键盘

输入：玩家的按键行为

输出：键盘发送的扫描码

处理过程：键盘按下按键发送的按键编码称为为扫描码。每个按键在按下和释放的时候都会产生不同的扫描码，分别称为通码和断码。

### 3.2 VGA显示器

输入：水平同步信号，垂直同步信号，像素的RGB值

输出：屏幕显示画面

处理过程：电子束从屏幕左上角开始，从左向右逐点扫描，每扫描完一行，电子束会回到屏幕的左边下一行的起始位置。每行用水平同步信号进行同步；当扫描完所有的行，形成一帧，用垂直同步信号进行同步，并使扫描回到屏幕左上方，开始下一帧。

### 3.3只读存储器(ROM)

输入：ROM地址

输出：数据

处理过程：从ROM中取出指定地址的数据并返回。

### 3.4随机存取存储器(RAM)

输入：RAM地址

输出：数据

处理过程：从RAM中取出指定地址的数据并返回。

## 4、软件模块设计

### 4.1状态更新器时钟(clk\_50Hz)

输入：板载的时钟信号clk

输出：50Hz的时钟信号update\_clk

处理过程：

1、创建寄存器cnt；

2、每遇到clk的上升沿，若cnt大于等于1000000，则清零，且切换update\_clk的状态；否则将cnt加一。

### 4.2 VGA工作时钟(clk\_25MHz)

输入：板载的时钟信号clk

输出：25MHz的时钟信号vga\_clk

处理过程：

1、创建寄存器cnt；

2、每遇到clk的上升沿，若cnt大于等于2，则清零，且切换vga\_clk的状态；否则将cnt加一。

### 4.3键盘输入驱动(keyboard\_driver)

输入：时钟信号clk，PS/2数据线ps2\_data，PS/2时钟线ps2\_clk，游戏状态state[1:0]

输出：游戏状态state[1:0]，方向键按键信号keys[1:0]

处理过程：

1. 检测PS/2数据线的信号，输出左右方向键的按键信号；
2. 若游戏状态为游戏开始前（0），且检测到任意键按下，则将游戏状态改为游戏进行中（1）；
3. 若游戏状态为游戏失败（2），且检测到任意键按下，则将游戏状态改为游戏进行中（1）；
4. 若游戏状态为游戏通关（3），且检测到任意键按下，则将游戏状态改为游戏开始前（0）；
5. 若游戏状态为游戏进行中，则进行以下操作：

* 若检测到左方向键按下，则在RAM中将角色的方向设为向左，keys[0]设为1；
* 若检测到右方向键按下，则在RAM中将角色的方向设为向右，keys[1]设为1；
* 若检测到左方向键松开，则将keys[0]设为0；
* 若检测到右方向键松开，则将keys[1]设为0；
* 若检测到跳跃键按下，且剩余跳跃次数大于0，则在RAM中将角色的垂直速度设为某个正整数值，垂直移动设为True，剩余跳跃次数减一；
* 若检测到射击键按下，则根据角色的当前位置和方向，在RAM中新增一个子弹实例，初始化其位置和水平速度，将子弹的数量加一。

6、无论处于何种游戏状态，若检测到返回键按下，则将游戏状态改为游戏开始前。

### 4.4初始化实例(instance\_initializer)

输入：游戏状态state[1:0]

处理过程：

1、若游戏状态从游戏开始前变为游戏进行中，则在RAM中进行以下操作：

* 初始化角色的位置，设置其方向向右，垂直移动为False，垂直速度为0，剩余跳跃次数为2；
* 初始化子弹的数量为0；
* 初始化各水果的位置、触发方向与触发状态；
* 初始化通关按钮的位置；
* 初始化死亡次数为0；

2、若游戏状态从游戏失败变为游戏进行中，则进行除了上述最后一条外的所有操作。

### 4.5位置更新(position\_updater)

输入：状态更新时钟update\_clk，游戏状态state[1:0]，方向键按键信号keys[1:0]

处理过程：每遇到update\_clk的上升沿，若游戏状态为进行中，则进行以下操作：

1. 若keys全为0或全为1，则不改变角色的横向坐标，否则使角色的横向坐标加上或减去一个固定值；
2. 从RAM中获取角色的垂直速度和垂直移动状态，若垂直移动状态为True，则根据其垂直速度改变角色的纵向坐标，并将垂直速度减去一个固定值，否则保持不变；

3、从RAM中获取每个子弹的位置和水平速度，据此改变其在RAM中的位置；

4、从RAM中获取每个水果的触发状态，若已触发，则根据其触发方向改变位置；若未触发，则根据角色的位置和水果的触发方向判断是否需要改变触发状态。

### 4.6碰撞检测(collision\_checker)

输入：状态更新时钟update\_clk，游戏状态state[1:0]

输出：游戏状态state[1:0]

处理过程：每遇到update\_clk的上升沿，若游戏状态为进行中，则进行以下操作：

1、从RAM中获取角色的位置，若与地面保持碰撞，则将垂直速度设为0，垂直移动设为False，剩余跳跃次数设为2。若未与地面碰撞，则将垂直移动设为True；

2、从RAM中获取每个水果的位置，若与角色发生碰撞，则将游戏状态设为游戏失败，并将死亡次数加一；

3、从RAM中button的位置，若与角色发生碰撞，则将游戏状态设为游戏通关。

### 4.7屏幕输出(render)

输入：VGA工作时钟vga\_clk，游戏状态state[1:0]

输出：水平同步信号hsync，垂直同步信号vsync，颜色数据rgb\_data

处理过程：每遇到clk\_20ms的上升沿，则进行以下操作：

1. 若游戏状态为游戏开始前（0），则渲染游戏开始界面；
2. 若游戏状态为游戏进行中（1），则根据RAM中的位置、动作等数据，依次渲染游戏背景、陷阱、子弹和人物；
3. 若游戏状态为游戏失败（2），则依次渲染游戏失败界面和死亡次数；
4. 若游戏状态为游戏通关（3），则依次渲染游戏通关界面和死亡次数。

## 5、进度计划

*（对进度计划进行介绍……）*

包含各检查点要求

## 6、人员与分工

*（对团队成员及分工进行介绍……）*