# 《Jack frost》课程设计

# 1、功能设计

## 1.1 概述

### 1.1.1 游戏玩法介绍：

"Jack Frost"是一款基于 FPGA 的冰雪主题游戏，灵感来自 Nitrome 游戏公司的经典作品。玩家将控制主角冰雪使者，通过将怪物和地形冰冻，以完成各种关卡的挑战。游戏的核心玩法为躲避怪物与冰冻地形，并包含各种解谜元素。每个关卡都有独特的地形和怪物设计，玩家需要巧妙运用冰冻技能克服各种障碍。

### 1.1.2 游戏制作模块说明：

1. 输入模块：

负责读取来自玩家输入设备（如键盘、操纵杆）的信号。这个模块需要处理玩家的移动方向、跳跃等命令输入。

2. 游戏逻辑模块：

游戏逻辑模块处理玩家动作，怪物行为、碰撞检测、地图生成、方块逻辑、关卡管理等底层逻辑。

3. 存储器模块：

存储游戏的静态资源，包括地图数据、怪物信息、关卡设计等。这可以是 FPGA 上的内部存储器或外部存储器。

4. 图形显示模块：

生成游戏画面，并将其输出到显示设备。这可以包括 FPGA 上的显示控制模块，用于处理图形渲染和输出。

### 1.1.2 制作关键点：

1. 实时性和性能优化：

由于 FPGA 资源有限，需要对游戏的各个模块进行性能优化，确保在 FPGA 上实现实时、流畅的游戏体验。

2. 输入处理和反馈：

实现对玩家输入的快速响应，并通过图形反馈，增强游戏的互动性。

3. 关卡设计与平衡：

设计各个关卡的地形和怪物分布，确保游戏难度在逐渐升级的同时能够保持平衡，提供

玩家良好的游戏体验。

4. 资源管理：

在 FPGA 上有效地管理游戏所需的资源，包括内部 RAM 的分配和外部存储器的读取。

5. 图形渲染：

通过有效的图形渲染技术，确保游戏画面在有限的硬件资源下呈现出良好的视觉效果。

在实施这个项目时，要密切关注硬件资源的使用情况，进行充分的测试和调试，确保游戏在 FPGA 上的稳定运行。

## 1.2 详细描述

### 1.2.1 游戏介绍：

《Jack Frost》由 Nitrome 游戏公司在 2007 年出品，是一款经典的 Flash 小游戏，因其优质像素风格和休闲特性，而受到众多玩家的好评。曾经在千禧一代的之间非常火热。

### 1.2.2 游戏公司介绍：

Nitrome 游戏公司是英国的一家游戏公司，专门制作像素游戏。经过十多年的发展，现在已经在全球拥有大量粉丝。虽然是大牌公司，但是做出来的游戏却是精致小巧，操作也简单，画质也精美，在线能玩。

近年来 Nitrome 开始向手机游戏产业转型，现已推出多款手机游戏，仍然沿袭了 Flash 时代的风格，画面由像素构成但十分精美，十分益智也十分有趣，操作简单难度适中，非常值得一玩。

本 Project 便是一次仿照 Nitrome 游戏公司经典的课题。

### 1.2.3 内容介绍：

1、游戏界面进入

游戏载入完毕后，点击“PLAY”，可选择单人、双人模式。选择关卡后即可开始游戏。

游戏开始界面

选择游玩模式

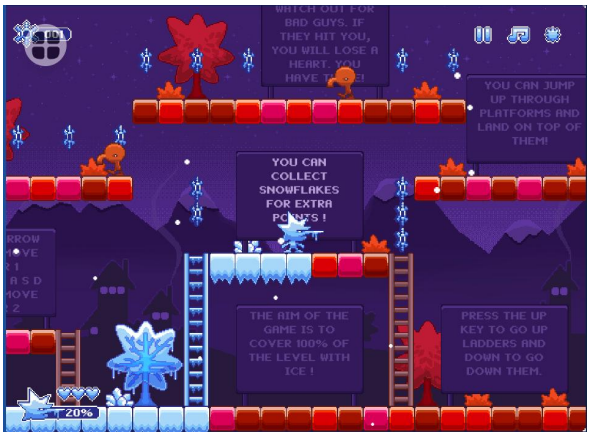

选择关卡


游戏会自动记录玩家游戏记录，保留玩家玩过的关卡，并在下一次启动游戏时，允许玩家开启已经通关的关卡。

2、游戏目的

玩家通过操作主角将画面中所有的可冰冻对象（包括地面、树、草和梯子）都变成冰。在进行冰冻的过程中，玩家需注意保持生命值，通过躲避，冰冻等手段规避怪物的攻击。

游戏界面



3、操作方法

玩家 1：←（左），→（右），↑（跳），↓(下梯子)；

玩家 2：A（左），D（右），W（跳），S(下梯子)。

跳到怪物头上会将它冻住


4、注意事项

1、由于游戏内容较多，读取游戏需一定的时间。

2、游戏中的怪物种类繁多，各种怪物有不同的能力，包括但不限于远程攻击、冰冻免疫等。因此要灵活应对。

3、在已经结冰的地面上，角色会因冰面影响导致行动不够灵活。

4、 每个关卡都设有雪花，吃掉后可获得更高的分数。

# 2、体系结构设计

## 功能设计：

1. 游戏启动和加载：

功能：加载游戏资源和关卡地图。

输入：启动信号。

处理：从存储器加载游戏资源。

输出：游戏初始化状态。

2. 玩家操作处理：

功能：接收玩家的操作指令。

输入：来自输入设备（例如键盘）的操作信号。

处理：解析操作指令。

输出：游戏内玩家角色的移动、跳跃等操作。

3. 游戏逻辑处理：

功能：处理游戏内的逻辑，包括怪物行为、碰撞检测等。

输入：玩家操作、游戏状态。

处理：根据游戏逻辑更新游戏状态。

输出：更新后的游戏状态。

4. 图形显示处理：

功能：生成游戏画面。

输入：游戏状态、地图数据。

处理：根据游戏状态和地图数据生成图形。

输出：图形显示到屏幕。

5. 关卡切换和记录：（记录进度实现可能有难度）

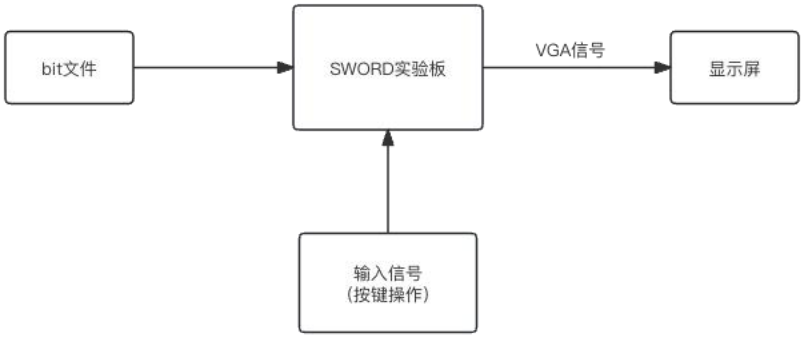
功能：处理关卡切换和记录玩家进度。

输入：游戏状态、玩家进度。

处理：根据游戏状态和玩家进度切换关卡，记录玩家的通关状态。

输出：切换后的关卡和玩家进度。

## 2.1 硬件体系结构




1. BIT 源文件：

由 ISE14.7 生成的 bit 文件可以将所有模块转化为实验板能解析的文件导入。

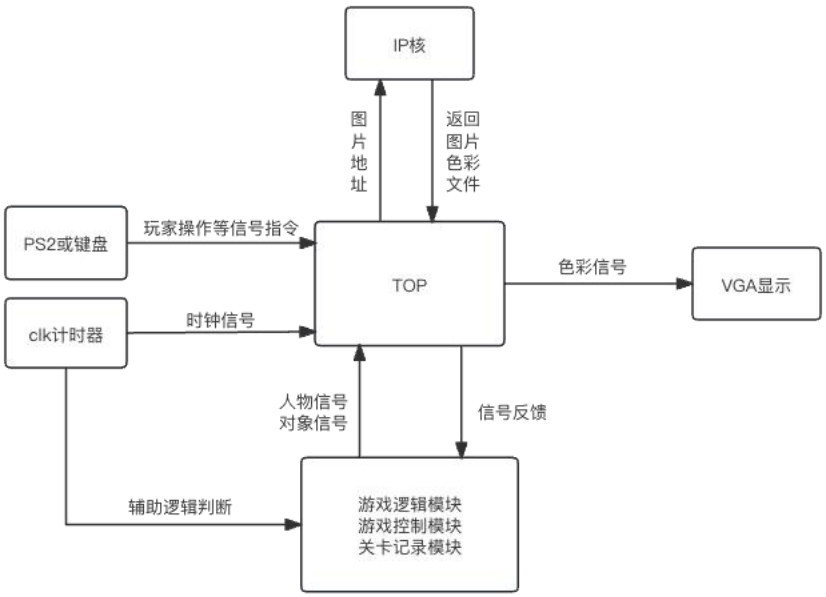
2. SWORD 实验板：

类似于 CPU\GPU 的作用，集成处理显示信号和逻辑信号，并将下一时刻的渲染文件信号通过 VGA 来存到显示屏。

3. 输入/输出接口：

包括与输入设备（按键、操纵杆）和输出设备（屏幕）的接口。这应该是基于 PS2 模块或者外接设备的属性状态。

## 2.2 软件体系结构



1. 输入/输出接口：

包括与输入设备（按键、操纵杆）和输出设备（屏幕）的接口。这应该是基于 PS2 模块或者外接设备的属性状态。

2. 存储器模块：

用于存储游戏资源，如地图数据、怪物信息、玩家角色等。此模块对提升游戏流畅性有极高作用。例如著名的马里奥，其游戏文件仅 40Kb，可保持流畅的运行。一部分就是因为其设计的资源调用函数能合理取用所显示的素材包，减轻游戏运行的负担。

3. 游戏控制模块：

负责游戏的启动、暂停、重启等操作。这个模块可能包括一些寄存器来管理游戏的不同状态。

4. 玩家操作处理模块：

解析并处理来自输入设备（主要是 PS2 键盘）的 xb 操作指令。可能需要一个防抖动模块或者连续点击判定信号的输入来保证人物的流畅移动。

5. 游戏逻辑处理模块：

处理游戏内的逻辑，包括怪物行为、碰撞检测等。例如人物运行加速度的完成、身下格子是否被踩到的判定、怪物触身的判定、景观物的变化判定，游戏结束判定等等。

6. 图形显示处理模块：

主要利用 VGA 模块来自动刷新存储器内的图片或者人物动作的显示，将所有素材均转为 coe 文件后多次调用 VGA 内核来显示屏幕

7. 关卡切换和记录模块：（该模块可能较难实现，原因在于每次重启游戏所有赋值将被刷新）

处理关卡切换和记录玩家进度。可能需要一个关卡管理器来跟踪游戏状态和关卡信息。

8. 时钟管理模块：

确保游戏的实时性和同步性。它可能包括一个时钟模块来同步游戏逻辑和图形显示，以及确保游戏运行在一致的时间基准上。

# 3、硬件模块设计

## 3.1BIT 源文件

输入：ISE 工程文件

输出：bit 流文件

处理过程：综合、Translate、Map、R&P 等过程处理

## 3.2SWORD 实验板

输入：bit 流文件、键盘输入信号

输出：VGA 色彩信号

处理过程：Sword 内部处理

## 3.3 输入/输出接口

不再赘述亲.

# 4、软件模块设计

## 4.1 游戏启动和加载

输入：启动信号

输出：游戏初始化状态

处理过程：从存储器加载游戏资源

## 4.2 玩家操作处理

输入：来自输入设备（例如键盘）的操作信号

输出：游戏内玩家角色的移动、跳跃等操作

处理过程：解析操作指令

## 4.3 游戏逻辑处理

输入：玩家操作、游戏状态

输出：更新后的游戏状态

处理过程：根据游戏逻辑更新游戏状态

## 4.4 图形显示处理

输入：游戏状态、地图数据

输出：图形显示到屏幕

处理过程：根据游戏状态和地图数据生成图形

# 5、进度计划

目前仅进行了项目计划和展望工作。

预计在冬 4 周前完成素材整理和熟悉系统以及理解 FPGA 底层逻辑，并在冬 6 周前完成基本的游戏实现。冬 7 周将完成项目报告，实现全部功能，修复项目漏洞并准备好项目展示。

# 6、人员与分工

王晓宇：代码 VGA 部分，制作视频

潘樾：代码 PS2 部分，撰写报告