1 概述

系统简述、软件设计目标、参考资料、修订版本记录

这部分论述整个系统的设计目标，明确地说明哪些功能是系统决定实现而哪些时不准备实现的。同时，对于非功能性的需求例如性能、可用性等，亦需提及。需求规格说明书对于这部分的内容来说是很重要的参考，看看其中明确了的功能性以及非功能性的需求。

这部分必须说清楚设计的全貌如何，务必使读者看后知道将实现的系统有什么特点和功能。在随后的文档部分，将解释设计是怎么来实现这些的。

T-Rex Runner Plus是以Google Chrome搭载的T-Rex Runner基础开发的跑酷冒险类单机游戏。该系统旨在满足各类用户的休闲娱乐需求。

玩家通过主菜单进入游戏、查看成就、退出游戏。进入游戏后，玩家使用键盘实时操纵小恐龙运动，系统对用户操作进行可视化反馈。系统在小恐龙的前方随机生成障碍物与道具并显示，进行小恐龙和障碍物与道具的碰撞判定，根据小恐龙与障碍物的碰撞情况扣除一定生命值，根据与道具的碰撞情况触发持续一段时间的特殊效果。当小恐龙生命值归0时，游戏结束。根据玩家的奔跑距离、道具获得情况、障碍物碰撞和消除情况设置成就，玩家可通过“查看成就”选项查看。

要求系统有效、快速、可靠、无误地完成上述操作。

本文档参考了Wayou Liu发布的T-Rex Runner源代码（<https://github.com/wayou/t-rex-runner>）。

本文档更新于2020年1月9日。

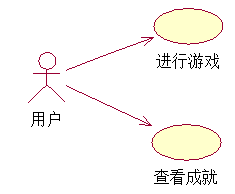
2 术语表

对本文档中所使用的各种术语进行说明。如果一些术语在需求规格说明书中已经说明过了，此处不用再重复，可以指引读者参考需求说明。

本系统：对T-Rex Runner Plus软件系统的简称。

3 用例

此处要求系统用用例图表述（UML），对每个用例（正常处理的情况）要有中文叙述。（设计人员对需求文档中的用例进一步确认和完善）



进行游戏是指玩家使用键盘控制小恐龙跳跃或下蹲，躲避障碍物并收集道具，通过奔跑距离或使用道具清除障碍物获得分数与成就的过程。

查看成就是指玩家在成就页面查看最高分数、已完成和未完成的成就的过程。

4 设计概述

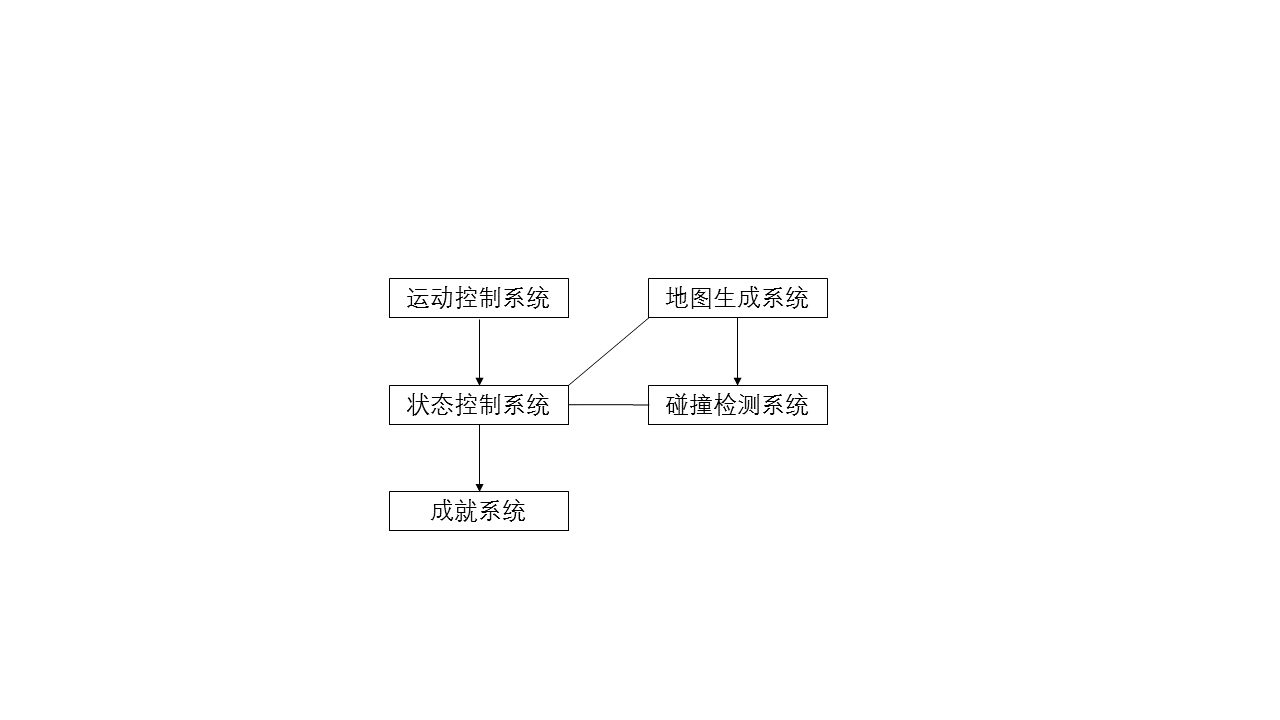
4.1 简述

这部分要求突出整个设计所采用的方法（是面向对象设计还是结构化设计）、系统的体系结构（例如客户/服务器结构）以及使用到的相应技术和工具（例如OMT、Rose）

系统设计采用面向对象方法，使用事件驱动模型。以下UML图使用Rational Rose绘制。

4.2 系统结构设计

这部分要求提供高层系统结构（顶层系统结构、各子系统结构）的描述，使用方框图来显示主要的组件及组件间的交互。最好是把逻辑结构同物理结构分离，对前者进行描述。别忘了说明图中用到的俗语和符号。



本系统分为5个子系统，分别为运动控制系统、状态控制系统、地图生成系统、碰撞检测系统和成就系统。运动控制系统处理玩家操作并转换为小恐龙的状态参数，传递给状态控制系统。状态控制系统根据小恐龙的状态和地图进行绘图，计算和判定小恐龙和地图当前状态，将小恐龙的位置参数传递给碰撞检测系统，将与成就有关的参数传递给成就系统。地图生成系统根据小恐龙当前里程生成地图上的障碍物与道具，并将物体参数传递给状态监测系统和碰撞检测系统。碰撞检测系统根据小恐龙和地图状态判定小恐龙是否与地图元素相碰撞，将碰撞信息传递给状态控制系统。成就系统处理和显示玩家达成的成就。

4.3 系统界面

各种提供给用户的界面以及外部系统在此处要予以说明。如果在需求规格说明书中已经对用户界面有了叙述，此处不用再重复，可以指引读者参考需求说明。如果系统提供了对其它系统的接口，比如说从其它软件系统导入/导出数据，必须在此说明。

系统界面中包括：玩家控制的小恐龙，位于界面左侧。地图（道具和障碍物），占据整个系统界面，并不断向左侧移动，以达到小恐龙向右侧奔跑的视觉效果。状态栏，位于界面左上角，显示小恐龙的生命值和道具获得情况、剩余时长。分数栏，位于界面右上角，显示当前分数和最高分数。

4.4 约束和假定

描述系统设计中最主要的约束，这些是由客户强制要求并在需求说明书写明的。说明系统是如何来适应这些约束的。

另外如果本系统跟其它外部系统交互或者依赖其它外部系统提供一些功能辅助，那么系统可能还受到其它的约束。这种情况下，要求清楚地描述与本系统有交互的软件类型以及这样导致的约束。

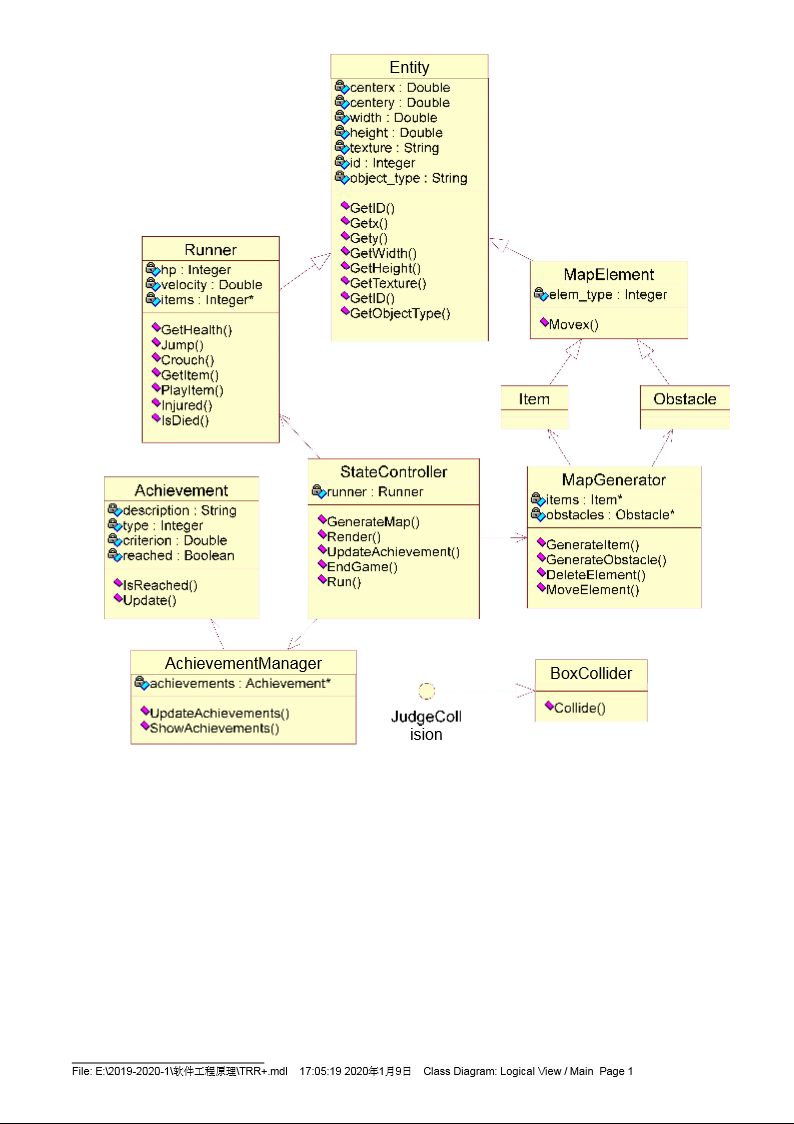
实现的语言和平台也会对系统有约束，同样在此予以说明。

对于因选择具体的设计实现而导致对系统的约束，简要地描述你的想法思路，经过怎么样的权衡，为什么要采取这样的设计等等。

暂无

5 对象模型

提供整个系统的对象模型，如果模型过大，按照可行的标准把它划分成小块，例如可以把客户端和服务器端的对象模型分开成两个图表述。在其中应该包含所有的系统对象。这些对象都是从理解需求后得到的。要明确哪些应该、哪些不应该被放进图中。所有对象之间的关联必须被确定并且必须指明联系的基数。聚合和继承关系必须清楚地确定下来。每个图必须附有简单的说明。



Entity类为地图上的实体，分为玩家可操控的Runner类和玩家不可操控的MapElement类，玩家不可操控的实体又分为道具Item类和障碍物Obstacle类。StateController是整个游戏进程的控制器，MapGenerator是不可操控的地图元素的生成者和维护者，AchievementManager类是成就的维护者。

6 对象描述

在这个部分叙述每个对象的细节，它的属性、它的方法。在这之前必须从逻辑上对对象进行组织。你可能需要用结构图把对象按子系统划分好。

为每个对象做一个条目。在系统对象模型中简要的描述它的用途、约束（如只能有一个实例），列出它的属性和方法。如果对象是存储在持久的数据容器中，标明它是持久对象，否则说明它是个临时对象（transient object）。

对每个对象的每个属性详细说明：名字、类型，如果属性不是很直观或者有约束（例如，每个对象的该属性必须有一个唯一的值或者值域是有限正整数等）。

对每个对象的每个方法详细说明：方法名，返回类型，返回值，参数，用途以及使用的算法的简要说明（如果不是特别简单的话）。如果对变量或者返回值由什么假定的话，Pre-conditions和Post-conditions必须在此说明。列出它或者被它调用的方法需要访问或者修改的属性。最后，提供可以验证实现方法的测试案例。

7 动态模型

这部分的作用是描述系统如何响应各种事件。一般使用顺序图和状态图。

确定不同的场景（Scenario）是第一步，不需要确定所有可能的场景，但是必须至少要覆盖典型的系统用例。不要自己去想当然地创造场景，通常的策略是描述那些客户可以感受得到的场景。

7.1 场景（Scenarios）

对每个场景做一则条目，包括以下内容：

场景名：给它一个可以望文生义的名字

场景描述：简要叙述场景是干什么的以及发生的动作的顺序。

顺序图：描述各种事件及事件发生的相对时间顺序。

7.2 状态图

这部分的内容包括系统动态模型重要的部分的状态图。可能你想为每个对象画一个状态图，但事实上会导致太多不期望的细节信息，只需要确定系统中一些重要的对象并为之提供状态图即可。

8 非功能性需求