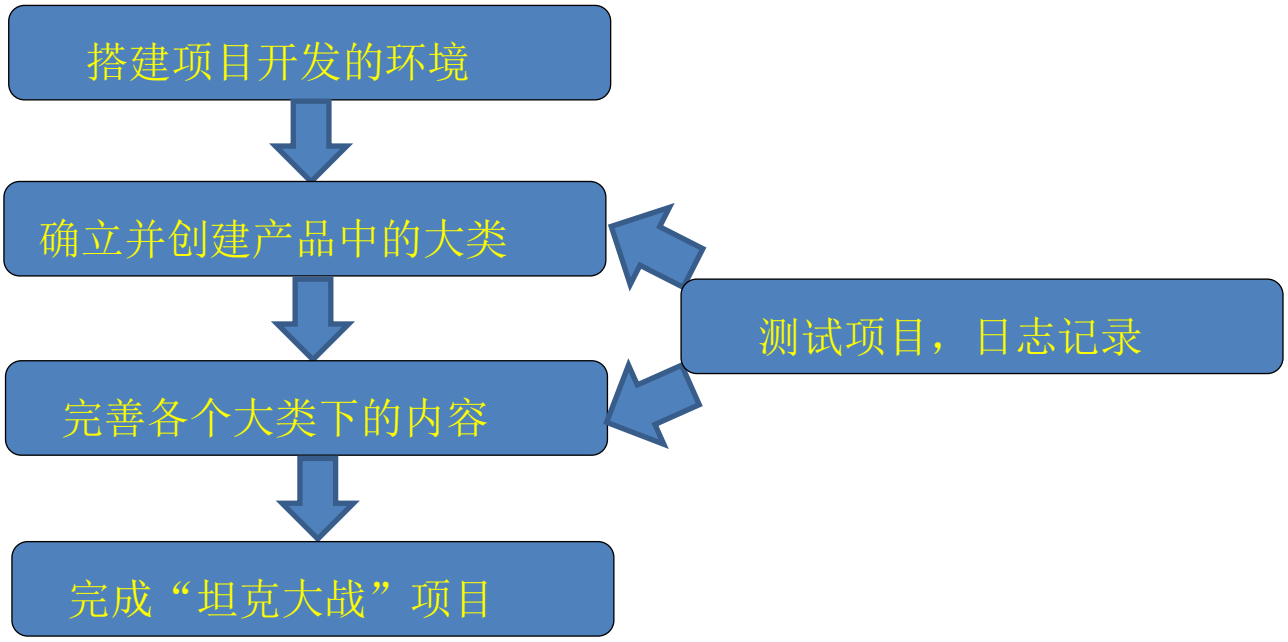
一、产品设计方案

1、产品目标：

通过 python 工具和面向对象的方法，重现经典的坦克大战游戏。游戏中将通过控制我方的坦克来摧毁敌方的坦克保护自己的“家”，把所有的敌方坦克消灭完达到胜利，而如果我方坦克被敌方坦克的子弹打中，游戏结束。

2、产品内容总策划（流程规划，设计与测试规范，开发日程表）：

<1>流程规划：



<2>设计与测试规范：

（1）命名：

○**模块**尽量使用小写命名，首字母保持小写，尽量不要用下划线(除非多个单词，且数量不多的情况)

○**类**名使用驼峰(CamelCase)命名风格，首字母大写，私有类可用一个下划线开头。

○**函数**名一律小写，如有多个单词，用下划线隔开。

○**变量**名尽量小写，如有多个单词，用下划线隔开。

○**常量**采用全大写，如有多个单词，使用下划线隔开。

○避免单字符名称，除了计数器和迭代器。避免包/模块名中的连字符(-)。避免双下划线开头并结尾的名称(Python保留，例如__init__)。避免使用小写字母 l(L)，大写字母 O(o)或 I(i)单独作为一个变量的名称，以区分数字 1 和 0。

（2）**注释**：

○单行注释和多行注释。

○在代码的关键部分(或比较复杂的地方)，能写注释的要尽量写注释。

○比较重要的注释段，使用多个等号隔开，可以更加醒目，突出重要性。

（3）**程序设计**：

○避免'裸代码'。

尽量不要直接将代码写在模块的顶层中，在执行主程序前应该总是检查 if __name__ == '__main__'，这样当模块被

导入时主程序就不会被执行。

（4）**测试**：（二选一）

系统测试是为了发现系统缺陷，保证产品质量而执行程序的过程，成功的测试是发现了至今尚未发现的错误的测试。测试的目的就是希望能以最少的人力和资源在最少的时间内发现潜在的各种错误和缺陷。应根据开发各阶段的需求、设计等文档或程序的内部结构，利用等价类、边界值、错误推断等测试方法，精心设计测试用例，并利用这些实例来运行程序，以便发现错误。系统测试是保证系统质量和可靠性的关键步骤，是对系统开发过程中的系统分析系统设计和实施的最后复查。

①人工测试：

【1】测试目标确保测试坦克大战小游戏各功能和效率测试范围坦克大战小游戏描述的所有功能、效率。

【2】采用技术与方法。使用等价类和边界值设计测试用例，使用动态黑盒测试方法对坦克大战小游戏描述的内容进行测试，核实以下内容：

○在使用有效数据时得到预期的结果。

○在使用无效数据时显示相应的错误消息或警告消息。

○软件配置项之间及软件配置项与硬件之间的接口正确。

○系统的输出及其格式正确。

○系统运行条件在边界状态和异常状态下，或在认为设定的状态下，系统功能和性能正确。

○系统访问和数据安全。

○系统的全部存储量、输入/输出通道和处理时间的余量正确。

○系统的功能、性能在强度测试下正常。

○设计中用于提高系统安全性、可靠性的结构、算法、容错、

○冗余、中断处理方案合理。

○对完整性级别高的系统，安全性、可靠性高。

○对有恢复或重置功能需求的系统，恢复或重置功能正确。

○对不同的实际问题外加相应的专门测试

②unitest 测试：

○人工解释结果。

○完全自动运行，而不需要人工干预。单元测试几乎是全自动的。

○自主判断被测试的方法是通过还是失败，而不需要人工解释结果。

○独立运行，而不依赖其它测试用例（即使测试的是同样的方法）。即，每一个测试用例 都是一个孤岛。

○每一个独立的测试都有它自己的不含参数及没有返回值的方法。如果方法不抛出异常而 正常退出则认为测试通过;否则，测试失败。

○为了编写测试用例，首先使该测试用例类成为 unittest

模块的 TestCase 类的子类。 TestCase 提供了很多你可以用于测试特定条件的测试用例的有用的方法。

○测试本身是类一个方法，并且该方法以 test 开头命名。

○所有测试样例放在项目根目录下的 test 文件夹中，如果你的所有测试都在一个文件中， 也可以省略文件夹，直接把这个文件放在根目录下，命名为 test_your_project.py（例 如 test_requests.py ）。

由于本贪食蛇小游戏操作等各方面都比较简单、也没有涉及到复杂的逻辑问题和往输入框中输入字符等问题，所以整个系统的测试需求比较简单，仅需点击查看其功能能否正常并且正确地完成即可，故采用的是纯手工测试。

<3>开发日程表：

日期	开发内容	解释
2020-6-10	搭建项目开发环境	安装 pycharm 并下载 pygame，logging 库。Pygame 库用以制作游戏主体，logging 库作日记
2020-6-11	分析并处理项目对象	设置了主类，坦克类，我方坦克类，敌方坦克类，子弹类，墙壁类，爆炸效果类和音效类。
2020-6-15	设置游戏窗	调整游戏窗口的大小和颜色，并新

	口，添加提示文字	增敌方坦克存活数量的文字提示，旨在创造良好的游戏体验。
2020-6-18	各大类对象的改进	1、加载我方坦克。 2、添加事件监听。
2020-6-19		1、随机生成敌方坦克。 2、敌方坦克随机发射子弹。 3、我方法子弹与敌方坦克的碰撞检测。
2020-6-21		1、添加爆炸效果。 2、我方坦克的消亡。
2020-6-23		1、子弹不能穿墙。 2、坦克不能穿墙。 3、双方坦克之间的碰撞检测。
2020-6-24	运行项目，测试主体程序	测试出游戏中可能存在的一些 bug 并尽量解决。

二、产品实现方案

1、系统的主要功能

<1>主类：主要包括开始游戏、结束游戏的功能。

```
class MainGame():
#开始游戏方法
    def startGame(self):
        pass
    def endGame(self):
        pass
```

<2>坦克类：主要包括坦克的创建、显示、移动及射击的功能。

```
class Tank():
        def __init__(self):
            pass
        #坦克的移动方法
        def move(self):
            pass
        #碰撞墙壁的方法
        def hitWalls(self):
        pass
```

<3>我方坦克类：继承坦克类，主要包括创建、与敌方坦克的碰撞方法。

<4>敌方坦克类：继承坦克类，主要包括创建、与我方坦克碰撞方法。

<5>子弹类：主要包括子弹的创建、显示及移动的功能。

<6>墙壁类：主要包括墙壁的创建、显示的功能。

<7>音效类：主要播放音乐。

2、关键技术和技术难点

<1>pygame 中的主要模块：

模块名	功能说明
pygame.display	访问显示设备
pygame.draw	绘制形状、线和点
pygame.event	管理事件
pygame.font	使用字体
pygame.image	加载和存储图片
pygame.key	读取键盘按键
pygame.mixer	声音
pygame.mouse	鼠标
pygame.music	播放音频
pygame.sndarray	操作声音数据
pygame.surface	管理图像和屏幕
pygame.sprite	操作移动图像

<2> Pygame 中常用的事件：

事件	参数	产生途径
QUIT	none	用户按下关闭按钮
ACTIVEEVENT	gain, state	激活或者隐藏 Pygame

KEYDOWN	unicode, key, mod	按下键
KEYUP	key, mod	放开键
MOUSEMOTION	pos, rel, buttons	鼠标移动
MOUSEBUTTONUP	pos, button	放开鼠标键
MOUSEBUTTONDOWN	pos, button	按下鼠标键

在 Pygame 框架中，MOUSEMOTION 事件会在鼠标动作的时候发生，它有如下所示 3 个参数。

□ buttons：一个含有 3 个数字的元组，3 个值分别代表左键、中键和右键，1 就表示
按下。

□ pos：位置

□ rel：代表现在距离上次产生鼠标事件时的距离和 MOUSEMOTION 类似，常用的鼠标事件还有 MOUSEBUTTONUP 和 MOUSEBUTTONDOWN 两个。在很多时候，开发者只需要知道鼠标按下就可以不用上面那个比较强大的事件了。这两个事件的参数如下所示。

□ button：这个值代表操作哪个按键

□ pos：位置

在 Pygame 框架中，键盘和游戏手柄的事件比较类似，处理键盘的事件为 KEYDOWN 和 KEYUP。KEYDOWN 和 KEYUP 事件的参数描述如下所示。

☐ key：按下或者放开的键值，是一个数字，因为很少有人可以记住，所以在 Pygame 中可以使用 K_xxx 来表示，比如字母 a 就是 K_a，还有 K_SPACE 和 K_RETURN。

☐ mod：包含了组合键信息，如果 mod&KMOD_CTRL 是真，表示用户同时按下了 Ctrl 键，类似的还有 KMODE_SHIFT 和 KMODE_ALT。

☐ unicode：代表了按下键对应的 Unicode 值。

<3>实现动画主要方法：

方法名	说明
pygame.image.load(filename)	加载一张图片
pygame.Surface.blit(source,dest,area=None, special_flags = 0)	将图片绘制到屏幕相应坐标上（后面两个参数默认，可以不传）
pygame.time.Clock()	获得 pygame 的时钟
pygame.time.Clock.tick(FPS)	设置 pygame 时钟的间隔时间

<3>双方坦克之间的碰撞检测：

如果我方坦克碰撞到敌方坦克，则我方坦克再不能继续移动。同理如果敌方坦克碰撞到我方坦克也不能继续移动。在我方坦克类中新增我方坦克与敌方坦克碰撞的方法。

```
class MyTank(Tank):
    def __init__(self,left,top):
        super(MyTank, self).__init__(left,top)
    #新增主动碰撞到敌方坦克的方法
    def hitEnemyTank(self):
        for eTank in MainGame.EnemyTank_list:
            if pygame.sprite.collide_rect(eTank,self):
                self.stay()
```

<4>坦克不能穿墙：

如果坦克与墙壁碰撞，则坦克不能继续移动，需要修改坦克的坐标为移动之前的。因此在坦克类中新增属性 oldLeft、oldTop 记录移动之前的坐标，新增 stay()、hitWalls()方法。

```
def stay(self):
  self.rect.left = self.oldLeft
  self.rect.top = self.oldTop
#新增碰撞墙壁的方法
def hitWalls(self):
    for wall in MainGame.Wall_list:
```

```
            if pygame.sprite.collide_rect(wall,self):
                self.stay()
```

<5>子弹不能穿墙：

子弹类中新增方法，子弹与墙壁的碰撞，如果子弹与墙壁碰撞，修改子弹的状态，墙壁的生命值减少，如果墙壁的生命值小于等于零时候修改墙壁的状态。

```
#新增子弹与墙壁的碰撞
def hitWalls(self):
    for wall in MainGame.Wall_list:
        if pygame.sprite.collide_rect(wall,self):
        #修改子弹的  live  属性
            self.live = False
            wall.hp -= 1
            if wall.hp <= 0:
                wall.live = False
```

<6>添加爆炸效果：

在我方子弹碰撞敌方坦克的方法中，如果检测到碰撞，产生爆炸类，并将爆炸效果添加到爆炸列表。

```
#新增我方子弹碰撞敌方坦克的方法
def hitEnemyTank(self):
```

```
for eTank in MainGame.EnemyTank_list:
    if pygame.sprite.collide_rect(eTank,self):
    #产生一个爆炸效果
        explode = Explode(eTank)
        #将爆炸效果加入到爆炸效果列表
        MainGame.Explode_list.append(explode)
        self.live = False
        eTank.live = False
```

3、已完成的改进和存在的问题

<1>墙壁数量和位置：

墙壁的位置错综复杂，保证了玩家应对挑战战略的丰富性。

```
def creatWalls(self):
    for i in range(3, 13):
        for j in range(1, 20):
            wall1 = Wall(130 * i, 200+130 * j)
            wall2 = Wall(845, 200+80 *j)
            MainGame.Wall_list.append(wall1)
            MainGame.Wall_list.append(wall2)
```

<2>初始化坦克位置，避免敌方坦克一出现就卡在墙里。

```
def creatEnemyTank(self):
    for i in range(MainGame.EnemTank_count):
        speed = random.randint(15, 25)
        # 每次都随机生成一个 left 值
        left = random.randint(1, 12)
        top = random.randint(0, 2)
        eTank = EnemyTank(left * 130, top * 80, speed)
        MainGame.EnemyTank_list.append(eTank)
```

<3>改进子弹速度和敌方坦克的灵敏值：

```
#敌方坦克速度
for i in range(MainGame.EnemTank_count):
    speed = random.randint(15, 25)
# 新增步数属性，用来控制敌方坦克随机移动
self.step = 10      #enemytank 类中类方法
def randMove(self):
    if self.step <= 0:
        self.direction = self.randDirection()
        self.step = 15
    else:
        self.move()
```

```
self.step -= 1
# 用以记录子弹速度（bullet 类中类方法）
self.speed=30
```

<4>待改进：敌我坦克初始位置如何不固定？在不固定的情况下，如何避免我方坦克的复活位置直接撞上敌方坦克或子弹路径？

三、测试大纲和测试报告

<1>测试大纲：

功能性，包括准确性、安全保密性、适合性、互操作性、功能依从性。

易用性，包括易学性、易理解性、易吸引性、易操作性、易用依从性。

可移植性，包括共存性、易安装性、适应性、易替换性、可移植的依从性。

可靠性，包括成熟性方面、易恢复性、容错性、可靠的依从性。

维护性，包括易测试性、易改变性、易分析性、稳定性、维护的依从性。

效率，包括资源利用性、时间特性、效率依从性。

<2>测试报告：

坦克大战小游戏已经通过测试，结果表明：

（1）功能性（适合性、准确性、互操作性、安全保密性、功能性依从性）

该软件各项功能运行正常，能够较准确的完成开始/停止游戏、加快速度、显示样式等基本操作。

（2）可靠性（成熟性、容错性、易恢复性、可靠性的依从性）

该软件对用户的误操作能较好的屏蔽，容错能力较好；软件在测试过程中极少出现异常退出，系统运行比较稳定；软件能较快的从失效状态重新启动，恢复到正常工作状态。软件遵循与可靠性相关的标准、约定或法规。

（3）可移植性（适应性、易安装性、共存性、易替换性、可移植性的依从性）

（4）易用性（易理解性、易学性、易操作性、易吸引性、易用性依从性）

该软件符合用户使用的要求，操作方便、易学、易理解、吸引用户使用；软件遵循与易用性相关的标准、预定、风格指南或法规。

（5）维护性（易分析性、易改变性、稳定性、易测试性、维护性的依从性）

该软件可以修改；修改后的功能可以开展测试；修改功能后

对其他功能不产生关联影响；能够较快的定位到缺陷并解决；软件遵循与维护性相关的标准或约定。

（6）效率（时间特性、资源特性、依从性）

该软件主要操作能够在 5S 内完成，系统资源使用正常，CPU 使用率平均为 30%以下，内存使用为 20%以下，I/O Wait 小于 2；软件遵循与效率相关的标准或约定。

（7）用户文档

用户手册对软件的主要功能和关键操作有相应的描述，易理解；用户文档描述和软件实际功能基本一致。

四、产品安装和使用说明

（1）产品安装：

解压文件，找到 tankall.py 并用 pychaem 进行打开，运行 pycharm 即可打开游戏。

（2）使用说明：

Q 键离开游戏，空格键控制坦克发射子弹，Esc 键使阵亡的己方坦克复活，paup 键向上移动，pgdn 键向下移动，home 键向左移动，end 键向右移动。我方坦克只有一滴血量。墙壁可破坏，其有三滴血量。尽量挑战己方一次不复活而全歼敌方坦克（疯人院难度）。