

信息安全工程课题研究

|  |  |
| --- | --- |
| 题目： | Slide CPATCH结题报告 |

|  |  |
| --- | --- |
| 成 员： | 龙东恒 宋文浩 许佳 谢贤彬 |
| 院 系： | 软件与微电子学院 |
| 导师姓名： | 孙惠平 |

二〇一七年 十二月

# **目录**

**[目录](#_Toc502262317)** [1](#_Toc502262317)

**[一、课题目标回顾](#_Toc502262318)** [2](#_Toc502262318)

**[二、](#_Toc502262319)****[项目过程概述](#_Toc502262319)** [3](#_Toc502262319)

[2.1 Slide CPATCHA 项目结构 3](#_Toc502262320)

[2.2 拼图处理概述 4](#_Toc502262321)

[2.3 前端Web技术概述 5](#_Toc502262322)

[2.3. 后台服务器技术概述 6](#_Toc502262323)

[2.3.1 服务器搭建 6](#_Toc502262324)

[2.3.2 数据库建立 6](#_Toc502262325)

**[三、项目过程细节](#_Toc502262326)** [7](#_Toc502262326)

[3.1 拼图处理 7](#_Toc502262327)

[3.2 前端页面 9](#_Toc502262328)

[3.3 后台服务器 9](#_Toc502262329)

[3.4 数据库 9](#_Toc502262330)

**[四、用户手册](#_Toc502262331)** [10](#_Toc502262331)

[4.1 源码获取及配置 10](#_Toc502262332)

[4.2 启动程序 11](#_Toc502262333)

[4.3 关闭程序 15](#_Toc502262334)

[4.4 生成拼图 16](#_Toc502262335)

[4.5 录入信息 18](#_Toc502262336)

**[五、解决方案评价](#_Toc502262337)** [19](#_Toc502262337)

[5.1 解决方案优势 19](#_Toc502262338)

[5.2 解决方案劣势 21](#_Toc502262339)

**[六 项目体会](#_Toc502262340)** [22](#_Toc502262340)

# **一、课题目标回顾**

随着互联网的不断发展和演进，以及社会对个人隐私的重视程度不断的加大，企业公司受到多方面的压力，纷纷使用各类办法保护内部数据，其中包含用户个人数据以及商业数据。当前市场出现了多种保护隐私数据的解决方案，最有效且低成本的方案是CPATCHA验证的使用。CPATCHA以其“Easy for human, hard for machine”的概念被广泛应用在区分人类和机器人行为的识别上，并且在多年的发展过程中，衍生出各类验证方案，如文本、汉字、图片选择及滑块等各式各样的验证码。在对国内市场中专做验证码市场的多个公司的产品进行比对后，小组提出自己的解决方案，该方案具有以下特点：

1. 更多用户交互。用户需要进行简单的三次判断、一次选择和一次拖动。
2. 降低识别风险。无知识型验证码本身不知识来挑战机器，所以机器无法通过学习以破解验证码（目前的方法是图像边界识别），此外新增的多个判断和选择较小的增大用户破解难度，但会急剧增大机器人破解的难度。从而降低了被机器人识别的风险。
3. 更好的兼容性。组内提出的解决方案及其落地的代码结果允许使用者可以通过简单的代码配置完成对此方案的动态配置，如确定子拼图选择个数，确定原图背景中出现的用于模糊视觉的假子拼图个数（0-1）。

综上，此次课题中，我们遵循软件开发中的敏捷开发模型，轻文档多交流，使用代码版本管理工具管理组员贡献的源码，并用市面中相对流行的前端、服务器、图像处理框架完成整个项目的源码编写。

# **项目过程概述**

## Slide CPATCHA 项目结构

验证码最广泛的应用场景是在PC端和移动端的浏览器中，鉴于普及性和一般性，本项目选择PC端作为项目实施环境，以常规开发模式，将项目分成前端和后台两部分单独完成。



图2-1 前后端分离结构

其中，前端由两个具有层级关系的HTML页面组成，服务器端有路由、渲染、数据库交互等功能组成，这类前后分离的开发模式具有以下特点：

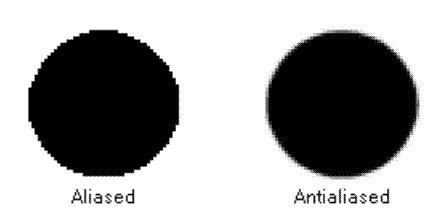
1. 解耦合。前端后台的源码分离，并选择不同的语言实施，可以有效地降低模块之间的耦合度，对某一模块进行修改时，产生的局部影响最小化。
2. 兼容性更强。由于前端技术的发展势头相当迅猛，用于前端渲染的框架层出不穷，就项目实施过程中，可供选择的框架除项目所使用的Django外，还有Flask、Web2.0等，此类框架都在Python的基础上，衍生出适应于不同场景的服务器开发框架。前后分离的模式，可以让后台的技术选型对前端的影响达到最小。
3. 职责分离。前端只负责业务逻辑的展示，不涉及业务逻辑的处理；后台只负责业务数据的传输和处理，不负责业务逻辑的展示。
4. 开发简便。过去，前端和后台共用一套语言开发（如java作为后台，jsp作为前端），对于熟悉HTML/CSS/JS的常规前端开发工作者需要去掌握新的语言（jsp），这增大了开发成本，而分离后，各自不再有学习的负担，能更耗的发挥自己的优势，让开发更简单。

综上所述，项目采取前后端分离的模式进行，各类工作由组员根据自己的喜好自由选择。

## 拼图处理概述

图片处理方面的代码，我们选取了Java语言，主要是使用AWT组件和Swing组件进行阴影区的添加与图片的剪裁等图片处理操作，得到了随机添加阴影区处理后的拼图背景以及剪裁好的拼图块。为了实现我们的设想，在可用性的基础上能够更加安全，一定程度上抵御边缘检测算法以及提高破解的代价，我们设计的拼图处理模块主要有以下几个特性：

1. 阴影区边界模糊与抗锯齿设计：使用setRenderingHint方法消除阴影区边界的锯齿，对阴影区边缘进行柔化处理，使阴影区边缘看起来更平滑，更接近实物的物体，机器更难以区分，处理前后对比效果如下图所示。



2. 阴影区颜色设计：为了使阴影区更自然，更难以被机器识别，系统会自动检测拼图中的随机位置的像素点，并将其颜色赋给阴影区。虽然在一定程度上牺牲了可用性且对图片的要求会更高，但这种设计带来的安全性的提高是显著的。

3. 双阴影区设计：为了弥补2中可用性的下降以及迷惑攻击者获取真实阴影区的位置，我们还设计了一个假的阴影区，假阴影区的颜色与真实阴影区颜色相同，实验表明这种设计极大地提高了可用性。

4. 阴影区不规则形状及随机旋转的设计：众所周知，规则形状（如矩形）会极大地降低破解的难度，因此我们使用Java绘图组件画了一个不规则的形状示意。为了进一步提高安全性，真假阴影区均被分别设计为随机旋转，若拼图块旋转不正确，即使真实阴影区位置正确，同样不能够验证成功，一定程度上提高了暴力破解的复杂度。

这些特性使得我们的滑动验证码设计比之市场上主流的验证码设计更安全，同时，假阴影区的设计保证了可用性。这种设计的最大的限制是对图片的要求比较高，不能使用有纯色区域或者是颜色单一的图片。

## 2.3 前端Web技术概述

前端Web技术选型中，选择通用前端开发语言，即HTML、CSS、JavaScript三件套。HTML和CSS在市场环境中不断更新换代，使开发者的开发体验不断增强，而与之配套的JavaScript也在努力前进，项目中选择JavaScript中的jQuery作为前端第三方插件库，以简化代码和实现部分动画效果。接下来对jQuery做简要的介绍。



图4-2 jQuery 3.0

jQuery使一个快速、简洁的JavaScript框架，使继Prototype之后有一个优秀的JavaScript代码库，其涉及宗旨是“Write less, Do More”，提倡写更少的代码，做更多事情，同大多数框架一样，简化开发过程中重复无用的工作，解放双手，创造未来。jQuery封装JavaScript常用的功能代码，提供一种简便的JavaScript涉及模式，优化HTML Doc操作、事件处理、动画涉及及Ajax交互。其核心特性可归纳为以下几点：

1. 具有独特的链式语法和短小清洗的多功能接口；
2. 具有高效灵活的CSS选择器，并且对CSS选择器进行扩展；
3. 拥有便捷的插件扩展机制和丰富的插件；
4. 兼容主流的各种浏览器。

此外，对于活跃于前端市场的单页渲染框架，如React、VUE等框架，由于其属于新型框架且在其国内缺乏相关的文档支持，使得学习成本急剧增大，严重影响开发进度，故弃之而选择常规三件套开发。

## 2.3. 后台服务器技术概述

后台的服务器中涉及简易服务器的搭建和数据库的建立，下面将对两者所采用的技术进行简要的说明。

### 2.3.1 服务器搭建

开

### 2.3.2 数据库建立

开

# **三、项目过程细节**

开

## 3.1 拼图处理

拼图处理的主要任务是选取图片、随机生成阴影区、记录随机坐标值以及剪裁拼图块。拼图处理的程序流程如下：

第一，图片选取高清图片，并将其统一处理为像素800\*600。由于我们设计上的限制，图片图片的要求比较高（高清、不能使用有纯色区域或者是颜色单一的图片）。

第二，生成坐标和旋转系数，要求真假阴影区不能重叠，阴影区与图片边界要有一定距离，旋转系数模拟为一位（即两种情况）。

第三，根据坐标画出阴影区形状，绘制形状通过内部类DShape模拟，共得到三种形状：真阴影区及拼图块的形状、假阴影区的形状、旋转后的拼图块的形状。

第四，拼图块的设计流程，分为两步，第一步将不规则形状之外的部分像素置零（设为透明），第二步沿着不规则形状边界剪裁拼图块成一个矩形并输出。

第五，阴影区的设计流程，阴影区的设计是我们的一大创新，也是本设计的核心要素，关于其安全性与可用性的设计已经在上述列出，尤其是阴影区颜色的设计和真假阴影区保持可用性的设计是我们的设计中的一大亮点。

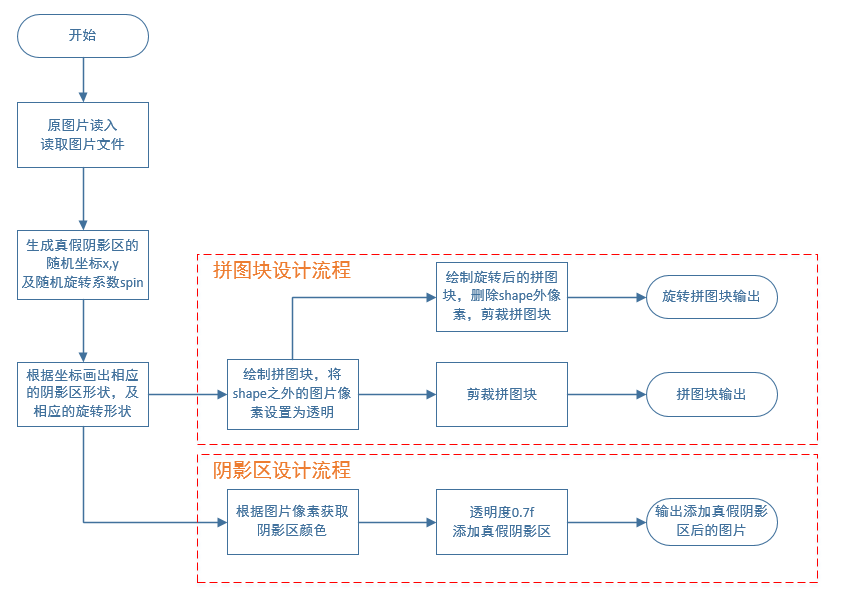


图1 拼图处理流程图

添加阴影区后的背景图片以及拼图块如下图所示。可见，我们的设计已经实现，并且有一定的趣味性，不过限制也非常明显如果没有假阴影区，真阴影区将非常难找。因此为了提高安全性，我们一定程度上牺牲了可用性，增加了认证的时间。

图2:添加阴影区后的背景图片（上：真阴影区；下：假阴影区）图3:拼图块 图4:旋转后的拼图块

## 3.2 前端页面

## 3.3 后台服务器

## 3.4 数据库

# **四、用户手册**

本节将介绍如何获取程序源码、环境配置及运行程序。

## 4.1 源码获取及配置

在浏览器中输入Github官网网址，在搜索栏内输入Slide-CPATCHA，第一条搜索结果就是当前项目的开源地址，点进去后，单击按钮Clone or Download按钮中的Download Zip，后续选择保存位置，并进行解压。图4-1为Slide-CPATCHA的开源网页截图。

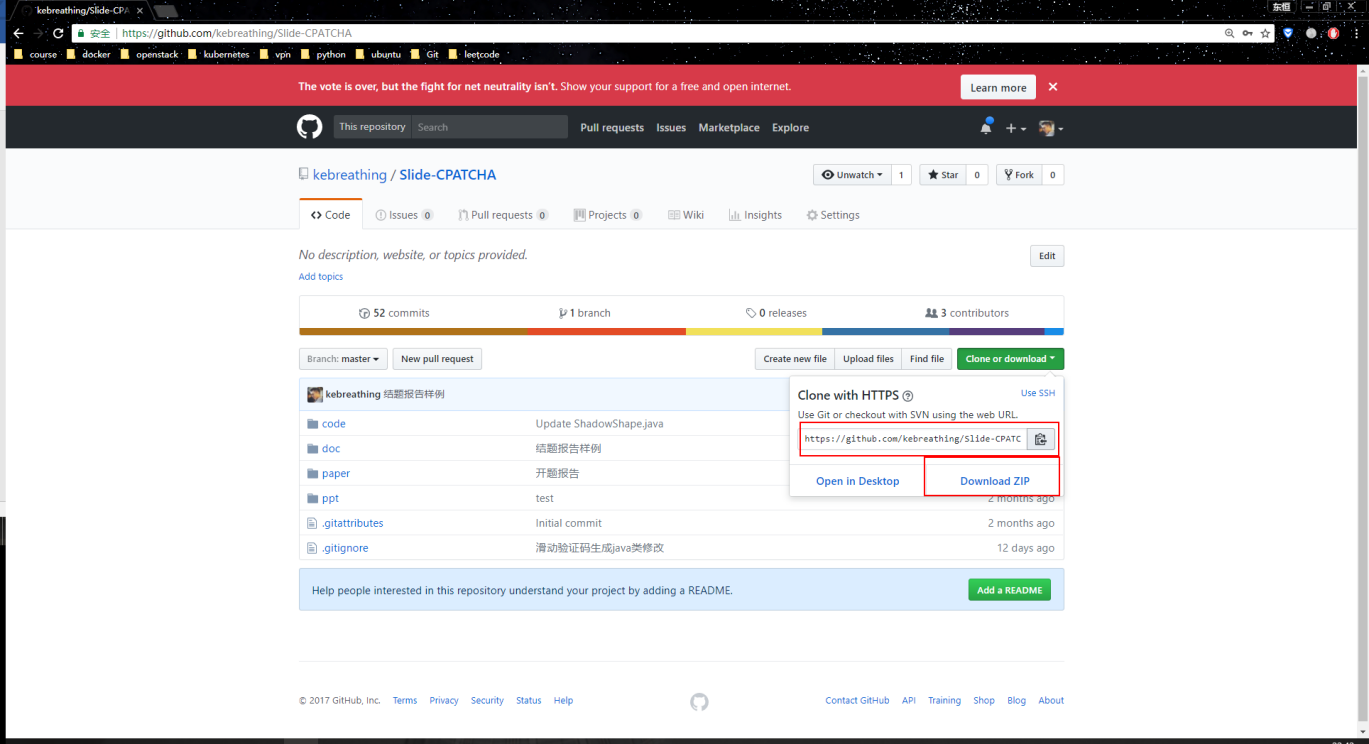


图4-1 Slide-CPATCHA 开源地址截图

解压后会看到目录中有以下四个目录：

* Code：存放组员贡献的源码，该目录下的子目录以组员名称为标识进行区分。其中的running为最终的整合源码，即将前端、后台、图像处理统一。
* Doc：存放同项目相关的文档，由于采用敏捷开发、项目规模小等原因，Doc中所以存有的文档只包含课题开题和结题报告。结题报告对项目的实施过程进行了相对详细的描述。
* Paper：存放同项目相关的文献资料。
* PPT：存放同课题相关的展示PPT，包括开题报告PPT和结题报告PPT。

了解基本的文件结构后，还需要对环境进行相关配置。以下罗列的项，是项目所依赖的包，务必需要正确安装并运行

* JAVA 8，图像处理/拼图生成是用JAVA实现的；
* Python 3，后台服务器由Python 3编写；
* MySQL，项目中涉及的数据都存储在MySQL中；
* 安装Python相关的包：
  + Flask
  + Json

环境配置完成后即可运行程序。

## 4.2 启动程序

开启命令行，进入/code/running目录，如图4-2所示，图中所使用的是windows系统下模拟Linux的命令行工具Cmder。

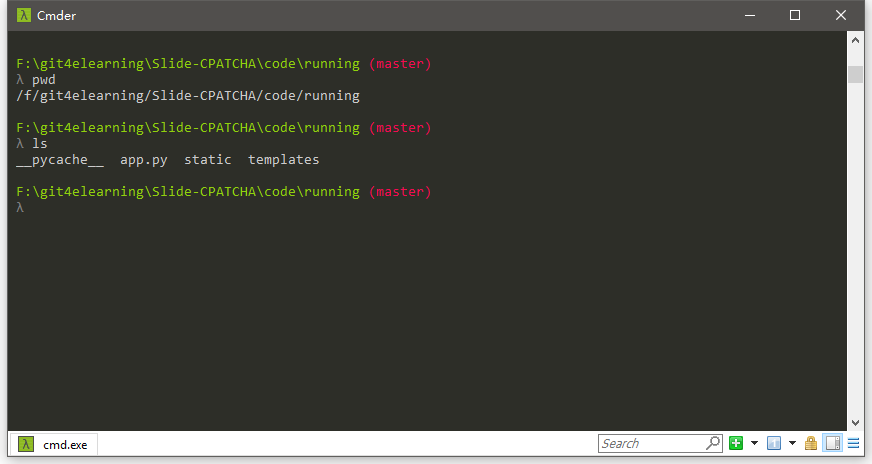


图4-2 进入/code/running目录

在命令行中敲入以下命令以设置Flask服务器的运行文件，如图4-3所示。注意Windows和Linux关于Flask运行文件的命令不同，前者用set，后者用export。再设置完成后，在当前目录下使用flask run 命令运行服务器，若无意外，命令行工具会显示服务器正在运行的IP和端口。默认为http://127.0.0.1:5000/。

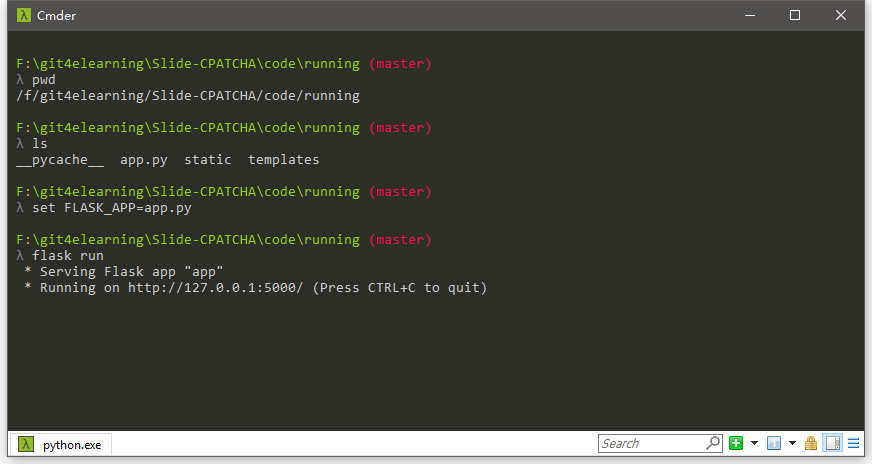


图4-3 Windows下运行Flask服务器

打开浏览器，输入服务器地址（http://127.0.0.1:5000/），添加index，即http://127.0.0.1:5000/index，进入验证码首页，如图4-4所示。

首页中会同时显示图库中6张已处理图片的原图，验证码的相关操作会基于每个框中显示的图片。且每次刷新页面显示的原图不会完全相同（取决于图库中所存有的图片数量）。点击各框内的Slide按钮，进入到详细的验证码拼图子页面，用户在子页面中进行验证码验证行为。

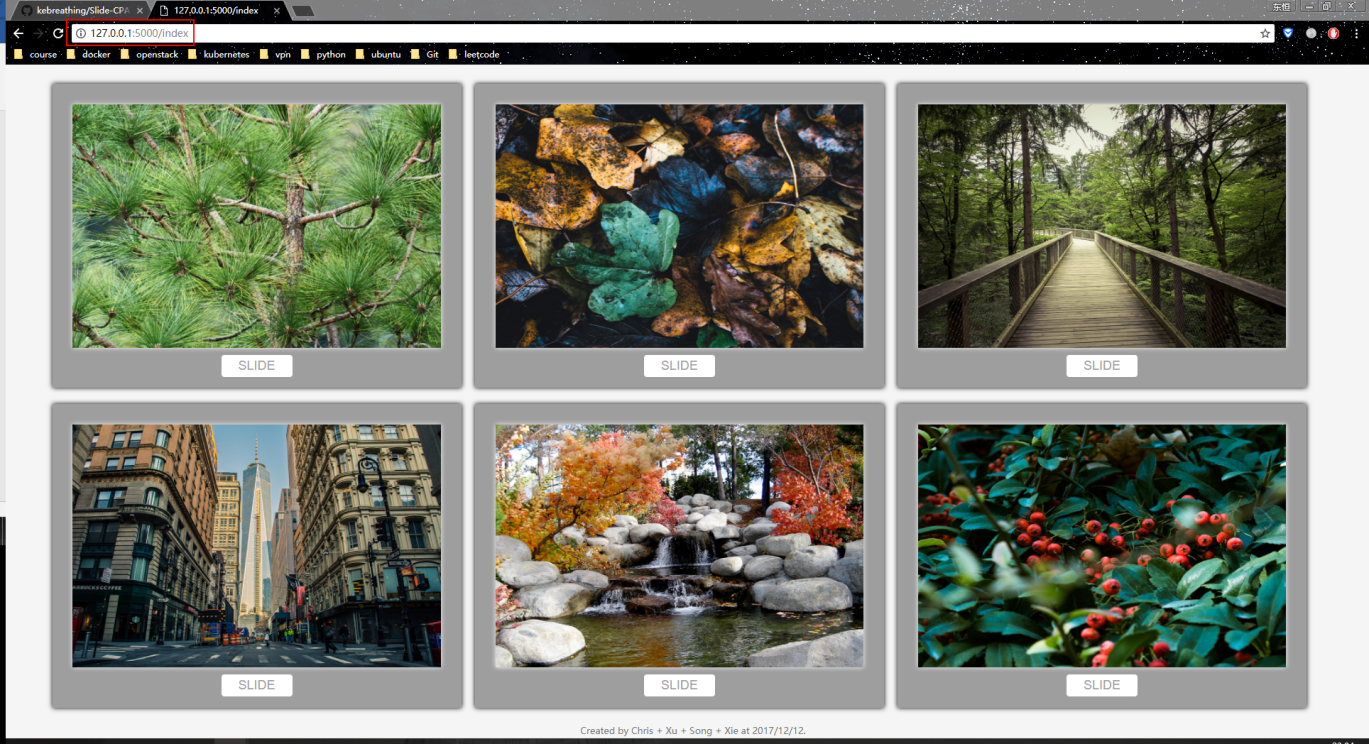


图4-4 验证码首页（第一次刷新）

拼图子页面如图4-6所示，由三部分组成，自左向右依次为：子拼图选择块，处理图块，鼠标信息记录块。



图4-6 拼图子页面

子拼图选择块。选择块内四个子拼图中的正确模块，下方较大拼图会跟随选择，以显示已选择的拼图，并在小子拼图上显示红色对号，如图4-7所示。

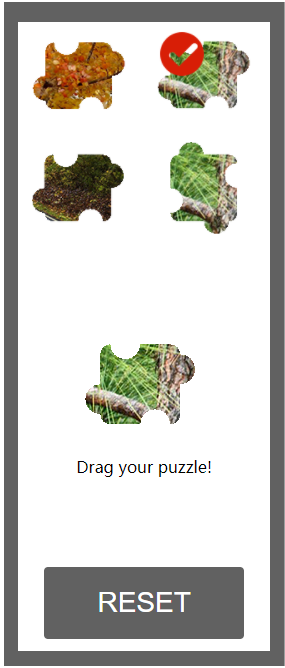
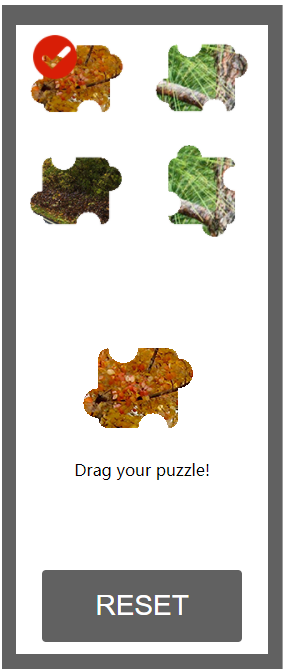


图 4-7 用户选择上方子拼图

选择正确的子拼图，拖动下方较大子拼图至处理图正确位置即可，具体操作如图4-8和图4-9所示。



图4-8 拖动前页面状态

拖动页面前如文档前面所述，处理图中有两处阴影，而左边子拼图是由其中一处阴影挖空生成的，用户需要选择正确的子拼图，并将其拖动至原本的位置才能完成验证。拖动完成后，左边子拼图红框部分将不再有子拼图存在。



图4-9 拖动后页面状态

当用户松开鼠标，即用户完成子拼图的拖动后，会向服务器发出验证请求，如图4-10所示。最终页面的右半部分显示整过过程中的鼠标行为情况。过程中，会记录鼠标选择子拼图的行为（点击与释放）、拖动子拼图的行为（拖拽开始和结束），以及鼠标在拖拽过程的坐标记录。

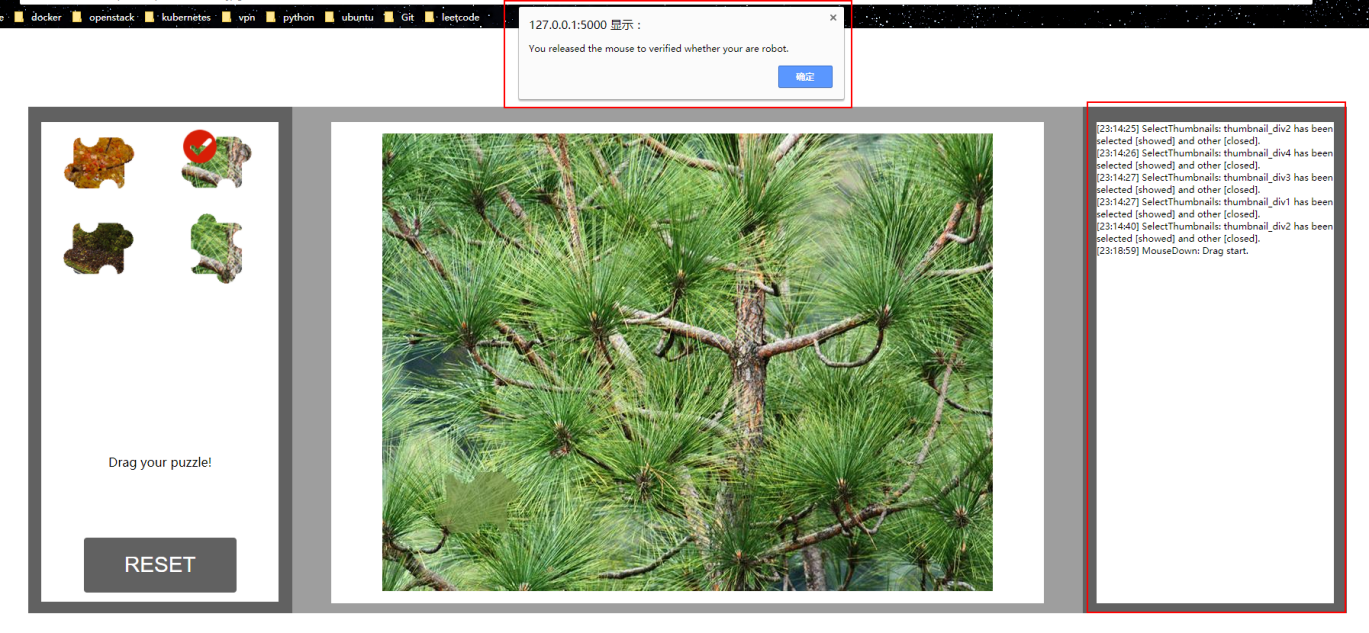


图4-10 鼠标行为

## 4.3 关闭程序

关闭应用程序，在刚才的命令行工具中，键入CTRL+C即可中止程序。注意，若修改过底层源码，一定要定期清理浏览器缓存。

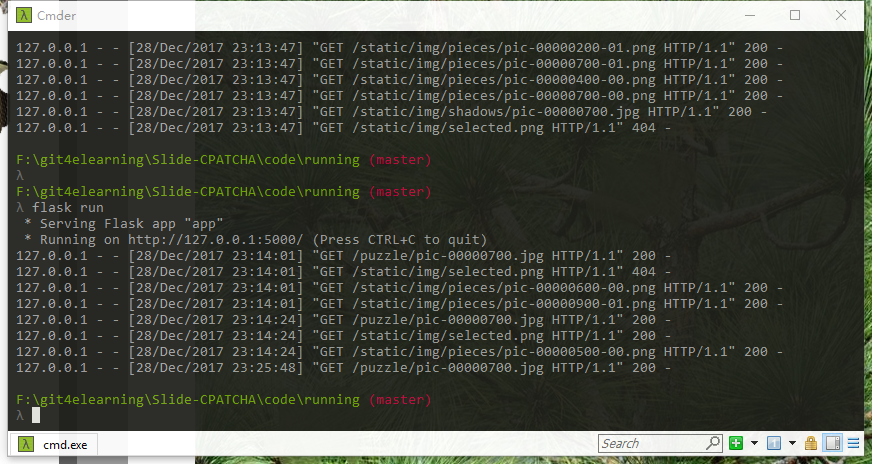


图4-11 中止程序运行

## 4.4 生成拼图

运行java项目GSPic，输入图片地址（例如本项目中原图地址\Slide-CPATCHA\code\running\static\img\pics\图片名.，阴影图片地址\Slide-CPATCHA\code\running\static\img\shadows\阴影图片名.，拼图块地址\Slide-CPATCHA\code\running\static\img\pieces\拼图块文件名.），程序自动运行读入图片并将相应的图片存入相应文件夹，十张示例图片如下图所示。

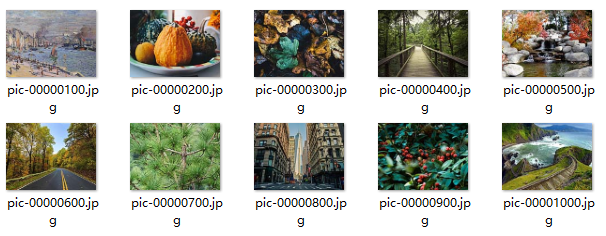


图1：pics文件夹中的原图示例

|  |  |
| --- | --- |
| pic-00001001 | pic-00000200 |
| pic-00000300 | pic-00000400 |
| pic-00000500 | pic-00000600 |
| pic-00000700 | pic-00000800 |
| pic-00000900 | pic-00001000 |

图2：shadows文件夹中的阴影区图片示例（红框：真阴影区；紫框：假阴影区）

|  |  |
| --- | --- |
| pic-00000100-1pic-00000100-2 | pic-00000200-00pic-00000200-01 |
| pic-00000300-00pic-00000300-01 | pic-00000400-00pic-00000400-01 |
| pic-00000500-00pic-00000500-01 | pic-00000600-00pic-00000600-01 |
| pic-00000700-00pic-00000700-01 | pic-00000800-00pic-00000800-01 |
| pic-00000900-00pic-00000900-01 | pic-00001000-00pic-00001000-01 |

图3：pieces文件夹中的拼图块示例（左：真拼图块；紫框：假旋转后的拼图块）

## 4.5 录入信息

此

# **五、解决方案评价**

项目在市面上已有的验证码产品上提出自己的新方案，在设计和实施的过程中，更多的考虑解决方案的安全性，而非产品的用户体验、性能等，换句话说，我们选择了安全性，以牺牲部分用户体验为代价。确实，解决方案离真正的产品上线还有很远的距离，若要使项目成为产品并真正上线，需要从以下方面入手：

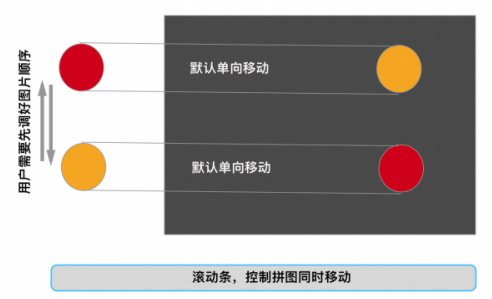
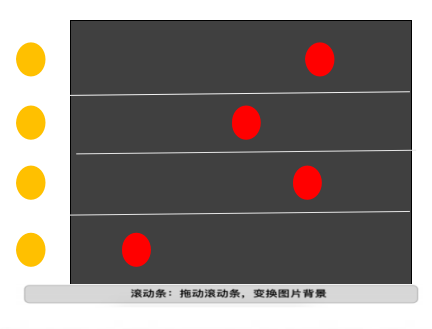
1. 产品应用场景：为谁而用？在哪使用？为PC浏览器设计，还是移动端App设计。是为常规百姓设计，还是为特殊人群（少数人）设计。
2. 产品核心特性。
3. 产品市场规模。
4. 扩展性。
5. 兼容性。
6. 合法性。
7. 识别风险强度：是不是真正的达到了易于人类，却难于机器人。

等等。虽然我们提出的解决方案还远不能达到产品上线程度，但在安全性上，我们提出了优于市面上已有验证码产品的特点。

## 5.1 解决方案优势

开题报告中，我们提及影响滑块验证码的因素有七点：拼图形状、拼图大小、图片背景中拼图选取位置、拼图挖去后空白处颜色填充、移动最小单位像素距离、移动过程速度变化、移动过程方向轨迹（具体内容请阅读开题报告文档）；并提出了四种方法：多滑块多拼图（依次拖动）、双滑块双拼图（滑块交换同时移动）、单滑块多拼图（多拼图形状不一但滑块形状唯一）、多滑块双拼图（最终方面，下文会详细介绍）如图5-1所示，四种方案的比较如表5-1所示。

在项目最终实施过程中，我们在开题报告种提出的方案四的基础上，进行改进，使得难度更大。



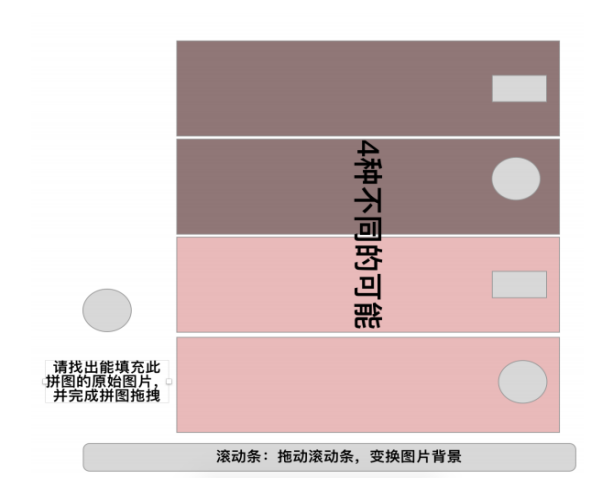


图5-1 开题报告中提及的四种方案

表5-1 四种方案对比

|  |  |  |
| --- | --- | --- |
| 方案 | 优点 | 缺陷 |
| 方案一 | 增强了对蛮力破解的抵御强度 | ➀对用户更不友好  ➁没有增强对其他攻击方式抵御能力 |
| 方案二 | ➀对用户来说更加快捷简单  ➁比常用的滑动验证更安全 | 在四种方案中安全性最差 |
| 方案三 | 安全性高，面面兼顾，能够有效地抵御各种攻击方法，尤其是蛮力攻击 | ➀设计太复杂，难度较高  ➁用户不友好，此方案平均每个用户都要多次点击转换按钮并分辨四张图，用户难以区分且增加了验证时间 |
| 方案四 | 既有方案三中用阴影区形状来辨别，又有判断拼图块区域，更安全更不易被破解 | ➀对一些图片，随机选取的阴影区用户会不好判断拼图块位置  ➁对图片有一定的要求 |

## 5.2 解决方案劣势

# **六 项目体会**