吉林大学 JILIN UNIVERSITY

本科生毕业论文（设计）

中文题目 面向教学的图灵机仿真系统的设计与开发

英文题目 Design and Development of a Teaching-Oriented Turing Machine Simulation System

学生姓名 杨明琅

学　　号 21200926

学　　院 计算机科学与技术学院

专　　业 计算机科学与技术专业

指导教师 郭东伟 教授

2024 年 5 月

面向教学的图灵机仿真系统的设计与开发

摘要

图灵机是一种抽象的计算模型，由英国数学家艾伦·图灵于 1936 年提出。它假设存在一种被称为“图灵机”的机器，能执行任何可能的计算。图灵机的概念对于理解计算机科学的本质有着深远的影响，是现代计算机科学的基石之一。本研究基于 Unity 工具设计并实现了一个图灵机仿真系统，并以 B/S 架构部署在云服务器，用户可以通过浏览器直接访问该系统。该系统可供学习者模拟图灵机的操作过程，深入理解和掌握计算理论的基本概念。该系统支持自定义图灵机规则，并提供可视化的操作界面，使得学习者能够直观地观察到图灵机的运算过程。同时在预设模式下提供了 2 标签系统、二元加法等算法的规则表。本文按照系统分析、总体设计、详细设计、部署与测试的一般软件开发流程，详细介绍了图灵机仿真系统的设计与开发过程。本系统的模块包括部件模型和用户界面模块、动作动画模块、输入响应模块、状态转移与逻辑控制模块、服务部署模块，在详细设计一章对这些模块的实现进行了详细说明。总体设计一章对系统的 UI 和对象进行了设计，其中对象主要为 InputProcesser 和 TuringMachine，分别是对输入响应处理器和图灵机的抽象。通过本文的介绍，读者不仅能够了解图灵机仿真系统的实现细节，还能够从中获得开发类似系统的经验和参考。相信这一系统将为广大学习者提供一个高效、便捷的学习工具，为计算机科学教育的发展作出贡献。

关键词：

图灵机；仿真系统；教学软件；Unity

Design and Development of a Teaching-Oriented Turing Machine Simulation System

Abstract

A Turing machine is an abstract model of computation proposed by British mathematician Alan Turing in 1936. It assumes the existence of a machine called a "Turing machine" that can perform any possible computation. The concept of Turing machine has had a profound impact on understanding the nature of computer science and is one of the cornerstones of modern computer science. In this study, we designed and implemented a Turing machine simulation system based on the Unity tool and deployed it on a cloud server in a B/S architecture, which can be directly accessed by users through a browser. The system can be used by learners to simulate the operation process of a Turing machine and gain a deeper understanding and mastery of the basic concepts of computational theory. The system supports customized Turing machine rules and provides a visual operation interface, which enables learners to intuitively observe the operation process of Turing machines. It also provides rule tables for algorithms such as 2-labeling system and binary addition in preset mode. This paper details the design and development process of the Turing machine simulation system following the general software development process of system analysis, general design, detailed design, deployment and testing. The modules of this system include component model and user interface module, action animation module, input response module, state transfer and logic control module, and service deployment module, and the implementation of these modules is described in detail in the detailed design chapter. The overall design chapter designs the UI and objects of the system, where the objects are mainly InputProcesser and TuringMachine, which are abstractions of Input Response Processor and Turing Machine respectively. Through the introduction of this paper, readers will not only be able to understand the implementation details of the Turing Machine simulation system, but also be able to gain experience and reference for developing similar systems. It is believed

that this system will provide an efficient and convenient learning tool for learners and contribute to the development of computer science education.

Keywords:

Turing machine;simulation system;educational software;Unity

目　　录

第 1 章 绪论

1.1 选题背景及意义

图灵机是艾伦·麦席森·图灵（Alan Mathison Turing，1912-1954）于 1936 年在论文《On Computable Numbers, With an Application to the Entscheidungsproblem》中提出的一种将人的计算行为抽象化的数学逻辑机。图灵在论文中并没有详细描述这种机器的物理结构，但从其逻辑功能可知，这种机器至少具有以下几个部件[1]：

1. 一条无限长的纸带 TAPE。纸带被划分为一个接一个的小格子，每个格子上包含一个来自有限字母表的符号，字母表中有一个特殊的符号用于表示空白。纸带上的格子从左到右依次被编号，纸带的右端可以无限延伸。
2. 一个读写头 HEAD。它可以在纸带上左右移动，可以读取并且擦写当前所指的格子上的符号。
3. 一张控制行为且数量有限的规则表 TABLE。它根据当前机器所处的状态以及当前读写头所指的格子上的符号来确定读写头下一步的动作，并改变状态寄存器的值，使机器进入一个新的状态。机器必须依据规则表中的规则来行动。
4. 一个状态寄存器。它用来保存图灵机当前所处的状态，图灵机的所有可能状态的数目是有限的。

一个精确的数学形式的图灵机定义为[10]：

一台图灵机是一个七元有序组（Q，Σ，Γ，δ，q_0，q_{accept}，q_{reject}），其中 Q,Σ,Γ 都是有限集合，且满足：

1. Q 是非空有穷状态集合；
2. Σ 是非空有穷输入字母表，其中不包含特殊的空白符□；
3. Γ 是非空有穷带字母表且 Σ⊂Γ；□∈Γ−Σ 为空白符，也是唯一允许出现无限次的字符；
4. δ:Q×Γ→Q×Γ×{L，R，−}是转移函数，其中 L，R 表示读写头是向左移还是向右移, −表示不移动；
5. q_0∈Q 是起始状态；

6. $q_{accept} \in Q$ 是接受状态；

7. $q_{reject} \in Q$ 是拒绝状态，且 $q_{reject} \neq q_{accept}$。

图灵机 M=（Q，Σ，Γ，δ，q_0，q_{accept}，q_{reject}）将以如下方式运作：

开始的时候将输入符号串 $\omega = \omega_0 \omega_1 \ldots \omega_{n-1} \in \Sigma^*$ 从左到右依此填在纸带的第 0，1，…，n−1 号格子上，其他格子保持空白（即填以空白符□)。M 的读写头指向第 0 号格子，M 处于状态 q_0。机器开始运行后，按照转移函数 δ 所描述的规则进行计算。例如，若当前机器的状态为 q，读写头所指的格子中的符号为 x，设 δ（q，x）=（q′，x′，L），则机器进入新状态 q′，将读写头所指的格子中的符号改为 x′，然后将读写头向左移动一个格子。若在某一时刻，读写头所指的是第 0 号格子，但根据转移函数它下一步将继续向左移，这时它停在原地不动。换句话说，读写头始终不移出纸带的左边界。若在某个时刻 M 根据转移函数进入了状态 q_{accept}，则它立刻停机并接受输入的字符串；若在某个时刻 M 根据转移函数进入了状态 q_{reject}，则它立刻停机并拒绝输入的字符串。

注意，转移函数 δ 是一个部分函数，换句话说对于某些 q、x，δ（q，x）可能没有定义，如果在运行中遇到下一个操作没有定义的情况，机器将立刻停机。

图灵机虽然只是一种逻辑机器，并非实际的物理机器，但其意义非凡。

图灵机为计算机科学和计算理论提供了一个基本的、抽象的数学模型。它清晰地定义了计算的概念，将计算过程抽象为一系列的符号操作和状态转移。这一模型不仅有助于我们理解计算机的工作原理，还为计算理论的研究奠定了基础。

图灵机帮助我们回答了“什么是可以计算的？”这一根本问题。通过图灵机，我们可以形式化地定义“可计算性”，即如果一个问题可以通过某个图灵机来解决，那么这个问题就是可计算的。这为计算机科学和数学中的许多领域提供了重要的理论基础。

图灵机也为计算复杂性理论提供了基础。复杂性理论关注于算法执行所需的时间和空间资源，而图灵机为我们提供了一个评估算法复杂性的框架。通过分析图灵机在解决特定问题时的行为和资源消耗，我们可以对算法的效率和性能进行评估。

对于拥有如此重要意义的图灵机，几乎任何计算机相关专业的学生在学习之初都不可避免地会接触到它的概念。但大部分涉及到图灵机的教材又都不会详细叙述图灵机，而只简单描述其工作原理并说明其历史意义，这也导致大多学生只知

“图灵机”之大名，而不知其为何物。这似乎是因为深入了解图灵机对现今学生并无实际意义，毕竟任何一台现代计算机的复杂性都远超图灵机，但反过来说，任何一台现代计算机其本质都是一台通用图灵机。从简单的图灵机模型出发，学生可以更好地理解计算机程序的设计和执行过程，也有助于培养学生的抽象思维能力和算法设计能力。

然而，由于其理论性和抽象性，学生对于图灵机的理解和掌握往往存在困难。这时，图灵机仿真系统的引入就显得尤为重要。另一点，根据学习科学研究，计算机工程学习者往往具有强烈的主动偏好，因此讲授驱动的教学方式对他们的激励作用较小。

图灵机仿真系统在教育中的作用主要体现在以下几个方面：

1. 直观性与可视化：图灵机作为一种理论计算模型，其操作原理对于初学者来说可能较为抽象和难以理解。仿真系统通过图形化界面和动态演示，将图灵机的内部结构和运算过程直观地展示给学生，使他们可以清晰地看到图灵机的各个部件如何协同工作，纸带上的读写操作如何进行，以及状态转移的具体过程。这种直观性和可视化大大降低了学习难度，有助于学生更好地理解和掌握图灵机的工作原理。

2. 辅助教学：图灵机仿真系统不仅是一个展示工具，更是一个教学辅助工具。教师可以通过系统演示图灵机的计算过程，将抽象的理论知识与具体的实践操作相结合，使学生更加深入地理解图灵机的运作机制。同时，教师还可以利用系统来讲解图灵机的相关概念和原理，提高教学效率和学生的学习效果。

3. 互动性与实验性：图灵机仿真系统具有良好的互动性和实验性，学生可以在系统中自行设计图灵机并进行实验。他们可以通过观察和分析图灵机的计算过程，深入理解其运算逻辑和工作原理。这种互动式的学习方式不仅激发了学生的学习兴趣和积极性，还培养了他们的实践能力和创新精神。

4. 个性化学习：每个学生对于图灵机的理解和掌握程度可能会有所不同。图灵机仿真系统允许学生根据自己的学习进度和能力水平进行个性化学习。他们可以通过反复实验和调整参数来加深对图灵机的理解，逐步提高自己的学习水平。

综上所述，图灵机仿真系统在教育领域的应用对于帮助学生更好地理解和掌握图灵机的工作原理具有重要意义。它不仅提高了教学效果和学习效率，还激发了学生的学习兴趣和积极性，促进了他们的个性化发展。

1.2 国内外研究现状

国内外对于图灵机的模拟研究已有不少先例，呈现出一定的活跃性。从实现形式上看，分为两类：一是以硬件形式实现，如陈凯设计了使用基础电子元件实现的“变色龙游戏”专用图灵机[2]，以及 Mike Davey 用微控制芯片、白色胶片、可擦写笔等材料实现的经典样式的图灵机[3]；一是以软件形式实现，如用 sendmail 仿真一个图灵机[4]，或者郭楠等人的专利——基于 Unity 的图灵机虚拟仿真系统[5]，以及国外 Andrew Hodges、Suzanne Britton 等各自实现的图灵机仿真软件。从实现内容上看，也可分为两类。或是实现了一些专用图灵机，有简单的如一元加法、简单判定问题、序列转换，有复杂的如二分搜索递归算法、0-1 背包动态规划算法；或是可由用户自定义状态转换表的图灵机框架。

在软件仿真图灵机方面，目前大多都只实现了抽象的图灵机模拟，即采用只用文字显示状态、纸带等信息的方法。此外许多研究要么只以文字展示，要么只给出了设计而没有实际的仿真软件。一个更为缺憾的点是可由用户自定义状态转换表的图灵机框架虽然可以视作通用图灵机，但其核心规则依托于现代计算机的机器语言，对于用户来说是不可见的。对于现有的图灵机仿真软件，允许用户在图灵机运行时更改各部件状态也是少见的。本研究致力于解决这些问题。此外，目前网络中可搜索到的图灵机仿真工具假定学习者已经掌握了基本概念。它们缺乏清晰的学习活动工作流程，无法指导新学员如何以及从哪里开始使用系统。这就使得新学员很难在系统中游刃有余。他们还依赖于高级数学和特异的用户交互。相反，我们的模拟器设计了清晰的学习活动工作流程，因此对于新用户来说，它易于使用和学习。

1.3 系统采用的相关技术

为了实现具有 3D 模型的图灵机仿真，本设计使用了 Unity 这一软件进行开发。Unity 技术与开发平台在建立图灵机 3D 模型方面具有显著优势，能够提供从建模到交互设计的全方位支持。

Unity 作为一款跨平台的游戏引擎，不仅在游戏开发领域大放异彩，其灵活的架构和丰富的 API 也为其他领域的 3D 模型构建提供了极大的便利。通过 Unity，

开发者可以轻松地创建复杂的 3D 场景，实现高精度的图形渲染，以及设计用户友好的交互界面。

对于图灵机的 3D 模型构建，Unity 提供了从基础建模到高级渲染的一系列工具。首先，利用 Unity 的 3D 建模工具，开发者可以根据图灵机的结构和工作原理，精确地构建出各个组成部分，如输入带、读写头、状态转换表等。同时，Unity 支持多种材质和纹理的导入，使得模型在视觉上更加逼真。

在交互设计方面，Unity 的丰富交互功能使得图灵机的操作过程得以生动展现。通过脚本编程，开发者可以实现模拟图灵机的工作流程，如输入字符、执行指令、状态转换等。此外，Unity 还支持多平台部署，包括 PC、移动设备、VR/AR 设备等，使得图灵机的 3D 模型可以在更广泛的场景下得到展示和应用。

在用户使用该系统的方式上，本设计将会采用 B/S 架构。使用 Unity 开发的软件可以以 WebGL 的形式发布。Unity 的 WebGL 构建选项允许开发者将他们的 3D 内容发布到网页浏览器，无需任何插件或额外的软件安装。

WebGL（全称为 Web Graphics Library）是一个 JavaScript API，用于在浏览器中渲染 2D 和 3D 图形，基于 OpenGL ES 2.0。WebGL 使用 GPU 来加速图形渲染，从而大大提高了网页的性能和响应速度。WebGL 的工作原理是在浏览器和图形硬件之间建立一个接口，使得 JavaScript 可以直接调用图形硬件进行渲染。通过 WebGL，开发者可以在网页上创建高质量的 3D 游戏、交互式数据可视化、虚拟地球仪、建筑可视化等应用程序。

采用 B/S（Browser/Server，浏览器/服务器）架构来设计系统意味着用户将主要通过标准的 Web 浏览器来与系统进行交互，而不需要安装额外的客户端软件。这种架构具有许多优点，包括易于部署、跨平台兼容性、易于维护和升级，以及较低的成本。

在 B/S 架构中，客户端只负责显示业务逻辑层传来的数据，并通过 Web 浏览器将用户的操作请求发送到服务器端。服务器端负责处理这些请求，执行相应的业务逻辑，并返回结果给客户端浏览器进行展示。因此，用户无需关心复杂的业务逻辑和数据处理过程，只需通过简单的浏览器界面即可与系统进行交互。

在 Unity 的 WebGL 构建中采用 B/S 架构意味着用户可以通过 Web 浏览器直接访问和体验 Unity 开发的游戏或应用，而无需安装 Unity 引擎或其他额外的客户端软件。这为用户提供了极大的便利性和灵活性，因为大多数现代设备都配备了

Web 浏览器，并且用户可以随时随地通过互联网访问系统。

第 2 章 系统分析

2.1 系统需求分析

本设计是一个使用 Unity 开发的图灵机仿真系统，其核心特点在于面向教学。针对该系统的面向教学特点，将该系统设计为包含有两种模式——预设模式和自由模式。

在预设模式下，图灵机的规则表、初始状态和结束状态都已经被设定好，依据规则表，允许输入的非空字符表事实上也已经被设定好，并且用户不能更改。该模式下，用户所能做的就是输入程序，构成程序的字符必须在允许的字符表中，否则在运行过程中会造成图灵机异常停机。用户输入的程序将会写入纸带，仿真的图灵机开始运行后，将会读取纸带上的字符，并依据当前状态和规则表进行动作。预设模式下的规则表预计将是一个能实现通用图灵机的小型规则表以及一些其他简单算法。该模式下，用户可以了解图灵机的基本运行原理，也能够学习计算机程序的运行原理，用于图灵机的程序的运行原理与现代计算机程序的运行原理是相通的。

在自由模式下，仅仅提供一个能够实现图灵机功能的框架，图灵机的核心——规则表由用户自定义，开始状态同样由用户自定义，用户可以根据需要来实现自己的专用或通用图灵机。自由模式下，用户还拥有更大的自由。当图灵机运行而程序未结束时，用户可以按下暂停，然后随意更改图灵机各部件的状态，包括状态寄存器、纸带上的字符和规则表中规则，当图灵机继续运行，将按照新的状态来进行动作。这样的自由可以使用户深入探索图灵机的运行原理、精确了解自己算法的优劣。

无论何种模式，首先要实现的是仿真图灵机的根本逻辑功能，即获取当前读写头所指向的纸带上的某个字符和当前状态寄存器中所存储的状态，然后根据当前字符及状态，在规则表中找到相应的规则，执行一系列动作，这些动作可能包含左移、右移、擦除当前字符与将纸带上的当前字符改写为其他字符，最后将状态寄存器中的状态转换为其他状态。

本系统还应该满足的用户需求是用户可以随时运行/暂停图灵机，可以调整图灵机的运行速度，这几个功能用于辅助用户更细致方便地观察图灵机的运行情况。另一个需要实现的功能是重启图灵机，让图灵机各部件回到最初的状态，自由模式下，重启不会初始化用户在图灵机运行时对规则表和纸带的更改。此外，用户可以

详细查看图灵机各部件状态的功能是不可或缺的。

当程序正常运行结束时，即图灵机停止时所处的状态为结束状态，应该弹窗提醒用户图灵机正常停机。当图灵机异常停机时，即图灵机遇到不在允许的字符集中的字符时或在规则表中找不到对应当前状态和字符的规则时，图灵机停止，若此时状态不是结束状态，应当提醒用户图灵机异常停机。

2.2 模块分析

本系统分模块设计，大致包含五个模块，详情可见图 2-1。

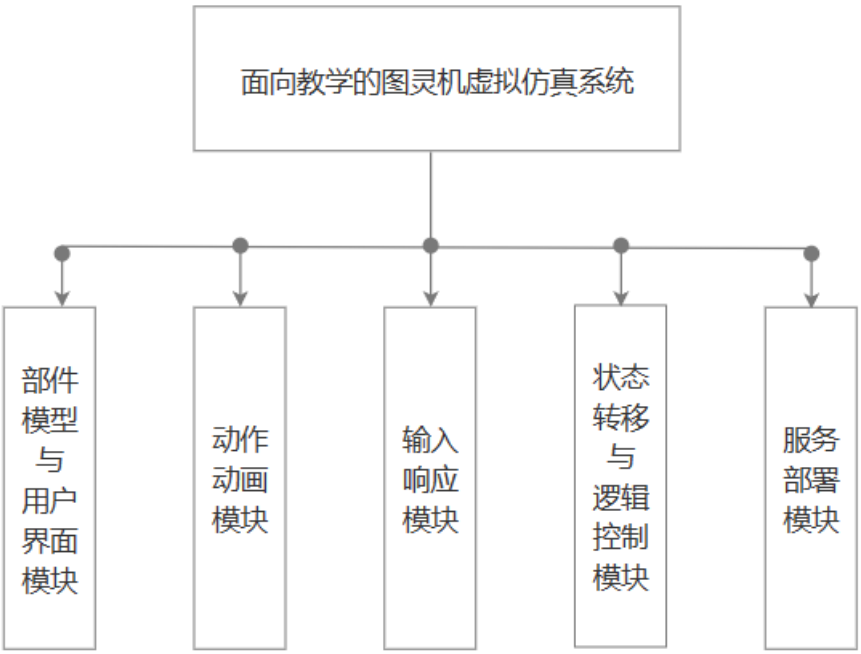

图 2-1 系统模块设计

其中，部件模型与用户界面模块包括图灵机各部件的建模和 UI 界面的样式、布局等；动作动画模块为图灵机运行时纸带、读写头移动以及状态寄存器变换状态时的动作动画的实现；输入响应模块即对用户键盘输入或鼠标点击进行响应，然后调用相应功能；状态转移与逻辑控制模块是图灵机仿真系统的关键模块，需要完成图灵机的主要逻辑功能与预设模式下的规则表的设计；服务部署模块则是将用 Unity 完成的项目打包为 WebGL 形式，并部署到服务器上，最终实现网页端访问系统的部分。

第 3 章 总体设计

3.1 UI 设计

本系统的 UI 遵循一般性的良好 UI 设计的原则，即主要为以下几点：

1. 用户友好性：UI 设计应该考虑用户的需求和习惯，确保用户可以轻松地理解和使用界面。界面布局应该清晰明了，操作流程应该简单易懂。
2. 一致性：UI 设计应该保持一致性，以便用户可以在不同的页面和功能之间无缝切换。一致性有助于提高用户的认知效率，减少用户的学习成本。
3. 可访问性：UI 设计应该考虑到不同用户群体的需求，包括残障人士、老年人等。设计应该遵循无障碍设计原则，确保所有人都可以轻松访问和使用界面。
4. 美观性：UI 设计应该注重美观性，以吸引用户的眼球并提高用户的满意度。设计应该符合当前的审美趋势，同时注重色彩、排版、图标等细节的处理。
5. 反馈性：UI 设计应该提供及时的反馈，以便用户可以清楚地了解自己的操作是否成功。反馈可以是视觉上的，也可以是声音或震动等形式的。

结合原则与本系统自身特征，即网页端访问，所作出的大致 UI 设计见图 3-1。

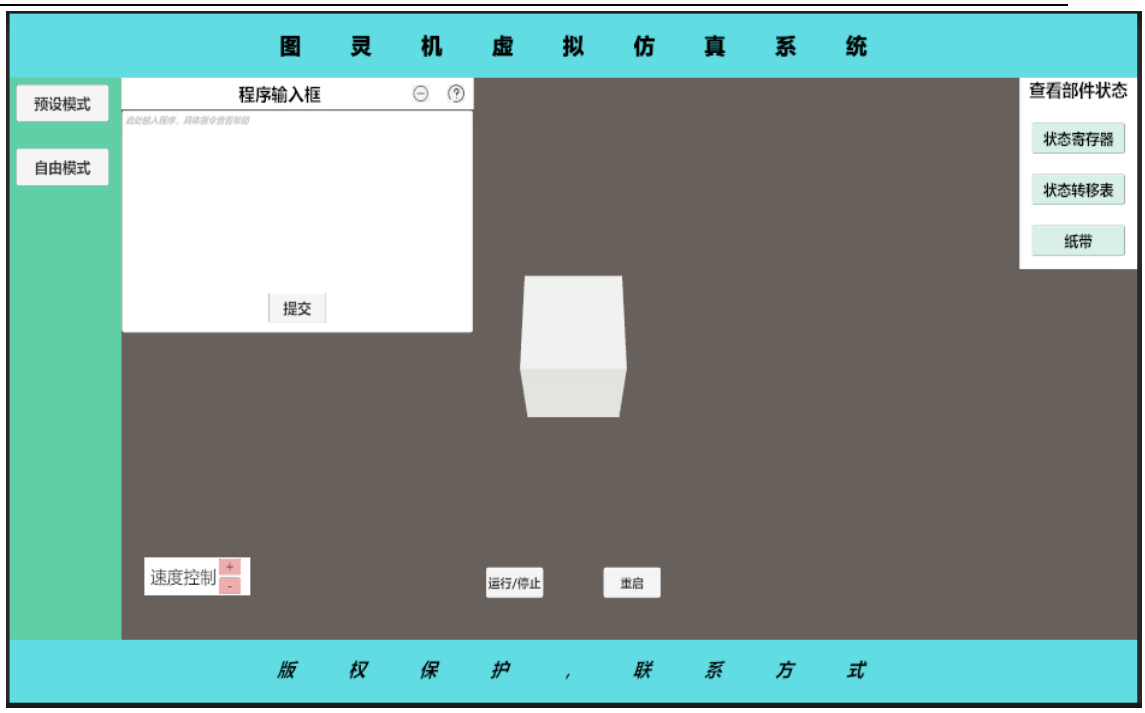

图 3-1 系统 UI 设计

在用户界面设计方面，本系统充分融合了用户友好性、一致性、美观性以及反馈性等多个核心要素，为用户打造了一个便捷、高效且直观的操作体验。

首先，考虑到用户友好性，我们针对本系统功能较少的特点，将所有功能以直观易懂的按钮形式直接展示给用户。用户无需在复杂的层级结构中寻找所需功能，只需通过简单的鼠标点击，即可轻松调用所需功能，极大地简化了操作流程，降低了使用门槛。

其次，我们注重用户界面的一致性。无论是预设模式还是自由模式，用户界面的设计都保持一致，区别仅在于用户操作权限的不同。这种一致性的设计让用户无需适应不同的界面风格，从而提高了用户的使用效率和体验。

在美观性方面，虽然受限于时间和技术水平，但我们仍然在不影响用户操作的前提下，力求打造简洁而富有美感的界面。系统整体采用青蓝作为主色调，给人以清新、自然之感。同时，我们选用了常见的图标样式，避免用户因图标过于独特而产生误解。在布局上，我们遵循突出中心的原则，将仿真的图灵机置于界面中心位置，周围则以清晰明了的布局围绕功能按键。非功能性信息提示则置于界面上下方，既不会干扰用户操作，又能在需要时提供方便的查看功能。

最后，在反馈性方面，我们设计了完善的提示机制。当系统出现某些状态变化时，如出错、图灵机停机等，系统会通过弹窗形式及时给予用户提示信息。这种反馈机制有助于用户及时了解系统状态，从而更好地进行下一步操作。

总之，本系统在用户界面设计方面充分考虑了用户友好性、一致性、美观性以及反馈性等多个因素，为用户提供了便捷、高效且直观的使用体验。我们相信，这一设计将能够满足用户的多样化需求，为用户带来更加舒适和愉快的使用感受。

3.2 对象设计

基于系统需求分析与模块分析，本系统主要涉及的对象有三个——UI 对象、用户输入处理器对象和图灵机对象。其中，UI 对象通过 Unity 自带的 uGUI 系统，组合基本元件，拼装完成即可。用户输入处理器对象首先需要 Unity 的事件系统组件来完成在输入框中输入文本的功能和对用户点击按钮的捕获，其次需要一个 InputProcesser 脚本来实现用户点击按钮后对各个功能的调用，将 InputProceser 中各个函数添加到对应的各个按钮的 Onclick 事件触发调用函数队列中即可。InputProceser 对象中涉及的函数见图 3-2。

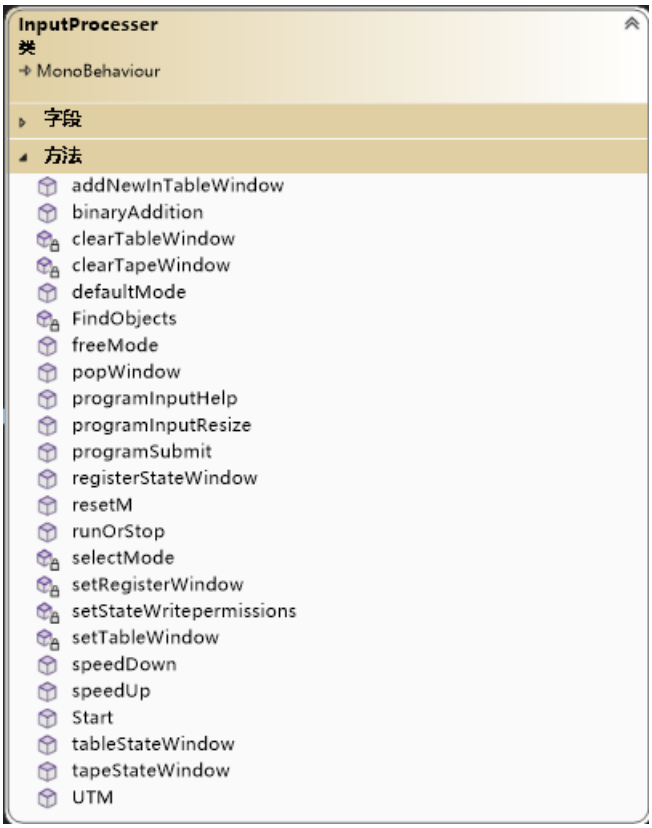

图 3-2 InputProcesser 类图

其中 Start 函数在整个程序运行之初调用一次，主要功能是选择模式为预设模式，其余各个函数显然地可以通过名字了解到它们的功能与对应按钮。

图灵机对象（TuringMachine）是对仿真图灵机的逻辑功能的抽象，仿真图灵机

的模型可采用 Unity 自带的一些简单几何体模型来组合完成。图灵机对象本身是由各个部件对象（Table、Register、Header、Tape）组合而成，其对外展示的功能基本通过调用各个部件的功能函数来完成。TuringMachine 的结构可见图 3-3。

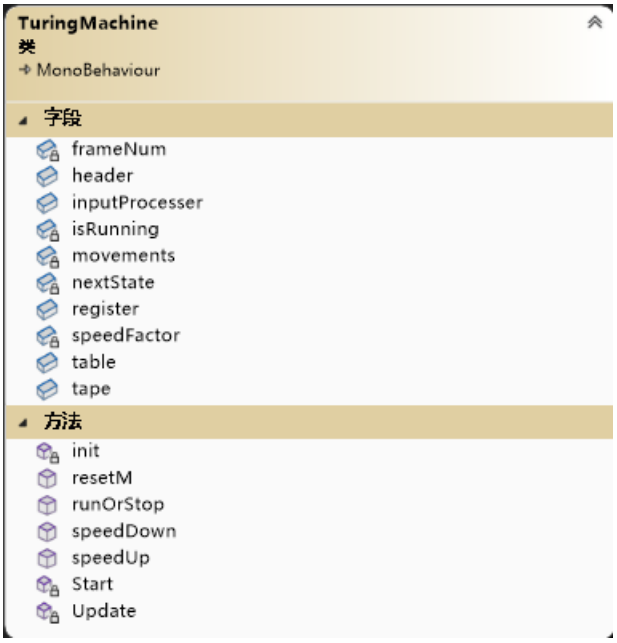

图 3-3 TuringMachine 的类图

TuringMachine 包含的主要函数有：

1. Init()，该函数根据各部件状态框中内容更新各部件对象的数据，其内部调用各个部件对象的 Init 函数。
2. RunOrStop()，该函数用于转换图灵机的运行状态，TuringMachine 内部有一项布尔型属性——isRunning，用于指示图灵机的运行状态，每次调用该函数则将 isRunning 取反。
3. ResetM()，该函数的功能为重启图灵机，内部首先调用 Init()，然后调用各个部件对象的 ResetM 函数。
4. Update()，该函数继承自 MonoBehaviour 类，在每一帧都会被调用一次。若图灵机是停止状态，该函数会调用 Init 函数；若图灵机是运行状态，则该函数会在特定帧让图灵机进行一次动作。动作由依据当前状态和当前字符从规则表中找到的对应转换项的动作队列指示，若动作队列中动作全部执行完毕，则图灵机转入新的状态，并获取新的转换项，直至停机。
5. SpeedUp()和 SpeedDown()，这两个函数用于控制图灵机的运行速度，图灵机的运行速度由属性 speedFactor 决定。处于运行状态的图灵机在特定帧执行一次动作，是否为特定帧其本质是在每一帧用累计帧数的数值对

speedFactor 作取余操作，若余数为 0，则为特定帧，否则不是。而每一帧所需时间基本固定，因此控制图灵机运行速度，只需要改变 speedFactor 属性的数值，其值越大，图灵机运行速度越慢，其值越小，图灵机运行得越快。运行速度预计设计了五档，增加或减小速度均不会超出这五档的限制。

接下来是对各个部件对象的设计，包括 Table、Register、Header 和 Tape。

Table 对象即图灵机中的规则表，该对象包含的主要属性有开始状态 startState、结束状态数组 endStates 和状态转换项列表 transferTable。TransferTable 是 Transfer 数据类型的 List 集合，Transfer 数据类型的结构为{ string currentState; char currrentChar; Movement[] movements; string nextState; }，即包括当前状态、当前字符、动作列表、下一个状态。Movement 类型是对图灵机动作的抽象，其结构为{ Command command; char c; }，Command 是对图灵机动作的枚举，包括 E(擦除)、P（覆写）、L（读写头左移）、R（读写头右移），char c 仅当动作为 P 时赋值，用来指示覆写后的字符。Table 对象向外公开的函数主要有两个，即 Init() 和 Convert(string currentState, char currentChar)。前者的功能是从规则表状态窗口获取规则表，并将赋值给 transferTable 属性。后者的功能是根据当前状态和当前指向字符，查找规则表，返回相应转换项，若未查找到，则返回 null。

Register 对象即图灵机中的寄存器，该对象仅包含属性 State，用于存储图灵机当前状态。该对象功能相对简单，包含函数有 Init()，从寄存器状态窗口获取状态值并存储；GetSate()，返回当前存储的状态；UpdateState(string state)，更新当前存储状态，包括寄存器窗口显示的状态值。

Tape 对象即图灵机中的纸带。其包含属性 initPos，用来指示纸带初始位置，在 Start 函数中被赋值。此外包含 GameObject 的 List 集合 cells，每个 cell 就是纸带上的一个小方格，对应一个长方体物体，物体上附有 text，即为方格中的内容。Tape 对象中含有的函数有 Init()，使用 getProgramFromTapeWindow()函数从 Tape 状态窗口获取用户输入程序，使用 generateCellsBy(string program)函数根据获取的程序生成 cells；LeftMove()和 rightMove()，分别控制纸带的左移和右移，使用 Unity 自带的用于控制物体移动的函数 transform.Translate(float,float,float)进行操作；updateAt(int index, char c)，用于更新指定位置的内容，index 指示位置，c 指示更新后的内容；getCharAt(int index)，用于获取指定位置的字符，index 指示位置；addCell()，在现有纸带末尾新增一个方格，方格内容为空（尽管理想的图灵机拥有无限长的纸

带，但限于实际，只能采用需要多少生成多少的方式）。

Header 对象即图灵机中的读写头，其拥有指向纸带对象的关联属性 tape 和指示自身在纸带中位置的属性 position，position 初始值为 0。该对象包含的主要函数有 getPosition()，返回读写头在纸带上所处的位置；getCurrentChar()，返回当前指向的字符；leftMove()和 rightMove()，用于控制读写头的左移和右移，因为设计上图灵机主体位置相对于视图不变动，因此读写头的左移和右移实际上分别调用纸带的右移和左移，另一点需要说明的是规则表中的动作左移 L 和右移 R 指的是读写头的移动；write(char value)，对应于图灵机动作 P，在读写头当前指向方格写下新值 value；erase()，对应于图灵机动作 E，即擦除读写头当前指向方格的内容，实际是将当前方格上的内容变为代表空白的特殊字符；resetM()，重启图灵机时读写头的变化。

以上即是对图灵机仿真系统的主要对象设计的说明，这些对象的设计旨在确保系统的功能性和用户友好性。UI 对象负责提供直观的用户界面，用户输入处理器对象处理用户的输入并转换为系统可理解的指令，而图灵机对象则负责执行这些指令并输出结果。

第 4 章 详细设计

在软件工程中，详细设计是软件开发的一个重要步骤，它是对总体设计的一个细化。详细设计阶段主要是基于需求分析的结果，设计出满足用户需求的软件系统产品。这包括对每个模块的具体实现进行详细规划，如算法的选择、数据结构的确定、接口的设计等。该部分主要是对 2.2 节涉及的前四个模块进行详尽设计说明。

4.1 部件模型与用户界面模块和动作动画模块

4.1.1 部件模型

图灵机中需要显性建模的部件包括纸带、读写头、状态寄存器以及负责控制的图灵机主体。图灵在文献[1]中所描述的机器结构大抵是受打字机的影响，该机器被图灵用来模拟人的计算过程并说明他的可计算性理论，但没有被实际制造出来。如果以机械结构建造单纸带的通用图灵机，其将是庞大且低效率的。尽管现代计算机本质上是一台通用图灵机，但物理结构与最初的图灵机描述大相径庭。本系统试图仿真的是在文献[1]中所描述的图灵机，并且注重的是图灵机的逻辑功能，而非物理结构，因此在物理建模方面对图灵机做了极致简化，这并不会妨碍用户对图灵机在逻辑上的运行原理的理解。

简化后的图灵机部件建模均为立方体，这可以通过创建 Unity 中预设的 Cube 模型来完成。为了使模型较为符合实际，对读写头做了（0.3，0.4，0.3）的缩放；对寄存器做了（0.6，0.4，0.1）的缩放，并且为了能较好地区分不同部件，将其材质颜色改为红色；纸带中的方格做了（0.5，0.05，0.5）的缩放，并将其保存为预制件，以便于在系统运行过程中创建方格。

4.1.2 用户界面

用户界面在 3.1 节已经作了一定说明，所有界面均使用 Unity 的 uGUI 系统制作完成。uGUI 是一套功能强大的用户界面（UI）开发框架，它为开发者提供了丰富的控件和布局选项，以便快速构建直观、交互性强的游戏和应用程序界面。该系统基于 Unity 引擎的内置渲染管道，与 Unity 的场景、光照、动画等核心功能无缝

集成，使得 UI 元素的渲染和交互更加高效和流畅。

uGUI 系统支持多种 UI 控件，包括按钮、滑动条、输入框、文本框、图像、布局容器等，开发者可以通过简单的拖拽和配置来创建和编辑 UI 元素。此外，uGUI 还提供了强大的事件处理机制，允许开发者为 UI 元素添加事件监听器，实现点击、滑动、拖拽等多种交互行为。

除了基本的 UI 控件和事件处理，uGUI 还支持自定义渲染和动画效果，开发者可以通过编写 Shader 和动画脚本来实现更加独特和个性化的 UI 效果。此外，uGUI 还支持多分辨率和屏幕适配，确保 UI 在不同设备和屏幕大小上都能够得到良好的展示效果。

本系统界面的基本控件包括图像、按钮、文本、输入框、滚动条，将这些控件组合拼装构成了整个页面。总体上看，页面分成了如下几个部分：Top、ModeBar、Bottom、ProgramInput、Status、FunctionBar。Top 置于页面顶部，为系统名称；Bottom 置于页面底部，包括版权保护和反馈方式等信息；ModeBar 位于页面左侧，放置了用户选择系统模式的功能按键；剩余部分为系统的核心区域。

核心区域左上角为 ProgramInput，用户在此处输入输入程序，点击提交后，程序将会复制到纸带上。该部分右上角有一个缩小/放大按钮，点击后可将程序输入框隐藏/显示；另有一个帮助按钮，点击后将显示帮助窗口，包含有关程序输入的格式等信息。核心区域右上角为 Status，该部分包含三个按钮——状态寄存器、状态转移表、纸带，点击后将分别使隐藏的对应状态窗口显示，这些窗口右上角有关闭按钮，点击后可使窗口隐藏。在自由模式下，用户可以在图灵机暂停状态下在这些窗口任意输入，以改变图灵机各部分状态。核心区域的正下方是 FunctionBar，该部分又可分为两部分。左侧为 Speed，用户在此点击按钮来改变图灵机运行速度；右侧为 RunBar，用户在此运行/暂停图灵机，并可重启图灵机。核心区域的中间即为仿真图灵机的模型，初始视角为侧俯视角。

4.1.3 动作动画

事实上，图灵机运行时的动作仅有四个，即 L、R、E、P，此外，还有一些图灵机显示状态的变化。但是，除了 L 和 R，其他的都可以通过简单地改变 Text 组件的文本内容来实现。对于 L 和 R，在 3.2 节已经提到，虽然代表的是读写头的左移和右移，但实际上改变的是纸带的位置，将其右移和左移来实现。纸带每次发生

移动动作，移动的距离为一个方格的距离，在 Tape 对象中以 cellWidth 属性来指示该距离大小。至于如何控制纸带的移动，我们可以使用 Unity 自带的函数 transform.Translate(float x, float y, float y)来改变物体的位置。在控制纸带移动时，我们首先需要获取纸带的 Transform 组件，然后通过调用 Translate 方法并传入相应的参数来改变纸带的位置。例如，如果我们要让纸带向右移动一定的距离，我们可以传入一个正数作为 x 参数。

4.2 输入响应模块

输入相应模块负责的是获取用户的键盘或鼠标输入，包括文本输入和按钮点击，然后对输入进行对应响应。对于本系统的输入获取，只需要添加 Unity 的 EventSystem 物体即可，而负责相应的是 InputProcesser 对象。InputProcesser 对象拥有的主要函数在 3.2 节已经进行过列举，下面对前文还未提及的函数的算法或步骤进行说明。

ProgramInputResize()，隐藏或显示程序输入框。InputProcesser 含有 GameObject 类型属性 programInputBox 与程序输入框物体相关联，函数首先判断 programInputBox 的状态，若为显示，则使用 SetActive(false)将其隐藏，否则使用 SetActive(true)使其显示。程序输入框状态改变时，按钮图标样式也会改变，使用 GetComponent<Image>()获取 Image 组件对象，并将 sprite 属性指向的精灵图改变即可。

ProgramSubmit()，将程序输入框中程序复制到纸带状态窗口中。该函数首先调用 clearTapeWindow()清除纸带状态窗口之前的值，随后获取程序输入框中存在的程序字符串，对于字符串中每一个非换行符的字符，根据 tapeWindow 中的 cellTemp 模板，生成一个显示字符的 cell。

ProgramInputHelp()、RegisterStateWindow()、TableStateWindow()和 TapeStateWindow()，这几个函数都用于控制隐藏的窗口显示，InputProcesser 对象中都有属性关联相应窗口物体，只需调用 GameObject 类的方法 SetActive(true)即可。

RunOrStop()，该函数调用 TuringMachine 对象的 RunOrStop 方法，而后者只是对 TuringMachine 对象中的 isRunning 属性简单取反。同理，ResetM()、SpeedUp()、

SpeedDown()都是调用 TuringMachine 对象的相应方法，那些方法在 3.2 节已有介绍。

DefaultMode()，系统进入预设模式。首先调用 setStateWritepermissions(false)将部件状态窗口的输入框设置于只读模式，然后调用 selectMode(defaultModeButton)变换模式按钮样式以明确系统此时所处模式，最后将设计好的初始状态和规则表注入到寄存器状态窗口和规则表状态窗口的内容中。

FreeMode()，系统进入自由模式。与 DefaultMode()类似，调用 setStateWritepermissions(true)和 selectMode(freeModeButton)分别设置输入框为可写模式和为模式按钮变换样式，随后清除各个部件状态窗口的所有预设内容，以让用户自己设计图灵机。

UTM()和 binaryAddition()，这两个函数用于在预设模式下选择算法。主要在其中以字符串数组的形式定义规则表和初始状态及结束状态，并将其传入 setTableWindow(string[,] transfers, string startState, string endState)以更改状态转移表状态窗口中的内容，该窗口内容会在图灵机停止时调用 Init()读取到 Table 对象中。

4.3 状态转移与逻辑控制模块

4.3.1 状态转移（规则表）

这一部分的内容是设计预设模式下的规则表，在 2.1 节系统需求分析中提到，预设模式下的规则表预计将是一个能实现通用图灵机的小型规则表。

对于通用图灵机的定义，图灵在其论文中这样给出：“It is possible to invent a single machine which can be used to compute any computable sequence. If this machine U is supplied with a tape on the beginning of which is written the S.D of some computing machine M , then U will compute the same sequence as M . In this section I explain in outline the behavior of the machine. The next section is devoted to giving the complete table for U .” [1] 换而言之，通用图灵机是一台能够模拟所有图灵机的图灵机。现代计算机是一台通用图灵机，其他图灵机的描述可以转变为符合现代计算机语言的程序，被现代计算机加载并运行计算。任何一段程序也可以说是一台图灵机的完整描述，甚至可以包括数据输入。

对于最小的定义，在 1956 年，香农提出了构造最简单通用图灵机的问题[6]。他考虑的是普通的确定性图灵机，有一个双向无限磁带和一个磁头。他提出用这台图灵机的指令数（即状态数 m 和磁带符号数 n 的乘积 mn）来衡量这台图灵机的复杂性，也可以考虑机器真正使用的命令数。

我们在下文中定义的通用图灵机模拟的是标签系统。对于正整数 m 和字母表($A = \{a_1, \dots, a_n, a_{n+1}\}$)，A 上的 m 标签系统对 A 上的单词$\beta$进行如下变换：删除$\beta$的前 m 个字母，并在结果的右侧附加一个取决于$\beta$的第一个字母的单词。这个过程重复进行，直到无法删除 m 个字母或第一个字母为a_{n+1}。形式上，我们有如下定义：

标签系统是一个三元组$T = (m, A, P)$，其中 m 是一个正整数，$A = \{a_1, \dots, a_n, a_{n+1}\}$是有限字母表，P 是一个映射关系，将$\{a_1, \dots, a_n\}$映射到字母表 A 上的有限词（字母序列）集合$A^*$上，将$a_{n+1}$映射到停止（STOP）上。标签系统$T = (m, A, P)$称为 m 标签系统，单词$\alpha_i = P(a_i) \in A^*$是标签系统 T 的产物，字母$a_{n+1}$是停止符号。

该系统也可描述为如下：

$$(T) \quad \begin{cases} a_i \to \alpha_i, & i \in \{1, \dots, n\} \\ a_{n+1} \to STOP \end{cases}$$

对标签系统的举例如下：

一个 2 标签系统T_1被定义为：$a_1 \to a_2 a_1 a_3, a_2 \to a_1, a_3 \to STOP$,对于初始单词$\beta = a_2 a_1 a_1$，$T_1$的计算过程为$a_2 a_1 a_1 \to a_1 a_1 \to a_2 a_1 a_3 \to a_3 a_1$。

Minsky 证明了通用 2 标签系统的存在[7]，因此，我们将只讨论 2 标签系统，它也具有如下性质：

1. 标签系统的计算只在以停止符a_{n+1}开头的单词上停止。
2. 产物$\alpha_i, i \in \{1, \dots, n\}$不为空。

以下所说标签系统将特指为 2 标签系统。

通用图灵机对标签系统的模拟如下。设 T 是一个在 A 上的标签系统，并且具有映射$a_i \to \alpha_i$。对于每一个字母$a_i \in A$关联一个正数N_i与代码A_i和$\widetilde{A_i}$（A_i可能等于$\widetilde{A_i}$），其形式为u^{N_i}（u重复N_i次），其中u是机器 U 的一串符号。即$a_i \to A_i(\tilde{A}_i) \to$

$u^{N_i}(\tilde{u}^{N_i})$。

代码A_i（或$\tilde{A}_i$）在 U 的磁带上用标记隔开。

对于$i \in \{1, \dots, n\}$，标签系统的映射$a_i \to \alpha_i = a_{i_1} a_{i_2} \dots a_{i_n}$被编码为$P_i = A_{i_n} A_{i_n - 1} \dots A_{i_2} A_{i_1}$。

将被标签系统转换的初始单词$\beta = a_r a_s a_t \dots a_w$，被编码为$S = A_r A_s A_t \dots A_w$（$S = \tilde{A}_r \tilde{A}_s \tilde{A}_t \dots \tilde{A}_w$）。

通用图灵机的纸带上的字符串初始形式为

$$Q_L \underbrace{P_{n+1} P_n \dots P_1 P_0}_{P} \underbrace{A_r A_s A_t \dots A_w}_{S} Q_R$$

其中，Q_L和Q_R分别是纸带左边和右边的无限部分，只由空白符号组成。P_{n+1}是停止符号a_{n+1}的代码，P_0是由几个字符组成的附加代码，用于分隔 P 部分与 S 部分。对于纸带双向无限的图灵机，其读写头初始位于 S 部分的左侧，并处于状态q_1。

假设 T 是任意标签系统，S_1和S_2是单词β_1和β_2的代码，并且有这样的关系：

$$\beta_1 \xrightarrow{T} \beta_2$$

那么通用图灵机 U 变换如下：

$$Q_L P_{n+1} P_n \dots P_1 P_0 S_1 Q_R \xrightarrow{U} Q_L P_{n+1} P_n \dots P_1 P_0 R S_2 Q_R,$$

（R 是被删除的前两个单元格的代码）。

UTM 的工作可分成三个阶段：

1. 在第一个阶段，UTM 搜索与代码A_r相对应的代码P_r，然后 UTM 删除代码A_r和A_s，也删除两者之间的间隔标记。
2. 在第二个阶段，UTM 将代码\(P_r\)按相反顺序写入纸带的\(Q_R\)部分。
3. 在第三个阶段，UTM 恢复自己的纸带，进行新一轮模拟。

标签系统的字母a_i对应的数字N_i具有这样一个特性，即每次模拟循环时，在代码P_r和代码A_r间有N_i个标记。在建模的第一阶段，UTM 的读写头会经过 P 部分的标记数等于代码A_r中符号 u 的数量。

在第一阶段之后，UTM 的纸带将会是

$$Q_{\mathrm{L}}P_{n+1}P_n\ldots P_{r+1}P_rP'_{r-1}\ldots P'_1P'_0R'A'_rA_sA_t\ldots A_wQ_{\mathrm{R}}$$

并且 UTM 的读写头位于$\mathrm{A'_r}$和$\mathrm{A_s}$中间的标记。然后 UTM 删除这个标记并开始模拟的第二个阶段。

经过第二个阶段，UTM 的纸带将会是

$$Q_{\mathrm{L}}P_{n+1}P_n\ldots P_{r+1}P''_rP''_{r-1}\ldots P''_1P''_0R''A_t\ldots A_wA_{r1}A_{r2}\ldots A_{rm_r}Q_{\mathrm{R}}$$

UTM 的读写头位于$\mathrm{P''_r}$的左侧，然后第三阶段开始。

经过第三阶段，UTM 的纸带将会变为

$$Q_{\mathrm{L}}P_{n+1}P_n\ldots P_1P_0RA_t\ldots A_wA_{r1}A_{r2}\ldots A_{rm_r}Q_{\mathrm{R}}$$

此时 UTM 的读写头位于 R 的右侧。

Yurii Rogozhin 提出了一种 UTM（24,2）的设计[8]。UTM（24，2）允许的字符为 0 和 1，状态为$\mathrm{q_i}(\mathrm{i}=1,2,\ldots,24)$。

标签系统的产物$\alpha_{\mathrm{i}}=\mathrm{a_{i1}a_{i2}}\ldots\mathrm{a_{im_i}}$的编码为

$\mathrm{P_i}=1010(00)^{\mathrm{N_{im_i}}}10(00)^{\mathrm{N_{im_i-1}}}\ldots10(00)^{\mathrm{N_{i1}}}10$，

且$\mathrm{N_1}=1,\mathrm{N_{k+1}}=\mathrm{N_k}+\mathrm{m_k}+2(\mathrm{k}\in\{1,\ldots,\mathrm{n}\})$。

即$\mathrm{A_j}=(00)^{\mathrm{N_j}},\mathrm{j}\in\{1,2,\ldots\mathrm{n}+1\}$，而 10 为间隔标记。

此外，$\mathrm{P_0}=10,\mathrm{P_{n+1}}=1110$。

将由标签系统转换的初始单词$\beta=\mathrm{a_ra_sa_t}\ldots\mathrm{a_w}$的编码 S 为

$\mathrm{S}=(01)^{\mathrm{N_r}}11(01)^{\mathrm{N_s}}11(01)^{\mathrm{N_t}}\ldots11(01)^{\mathrm{N_w}}$，

即$\widetilde{\mathrm{A}}_{\mathrm{i}}=(01)^{\mathrm{N_j}},\mathrm{j}\in\{1,2,\ldots,\mathrm{n}+1\}$，而 11 为间隔标记。

而该 UTM 的规则集见图 4-1，每条规则的格式为：当前状态-当前字符-覆写后的字符-左/右移-转换后的状态。

$q_1 00 R q_5$	$q_2 01 R q_1$	$q_3 00 L q_4$	$q_4 01 L q_{12}$
$q_1 11 R q_2$	$q_2 11 L q_3$	$q_3 10 L q_2$	$q_4 10 L q_9$
$q_5 01 R q_1$	$q_6 00 L q_7$	$q_7 00 L q_8$	$q_8 00 L q_7$
$q_5 10 L q_6$	$q_6 11 L q_7$	$q_7 10 L q_6$	$q_8 11 R q_2$
$q_9 00 R q_{19}$	$q_{10} 01 L q_4$	$q_{11} 00 L q_4$	$q_{12} 00 R q_{19}$
$q_9 11 L q_4$	$q_{10} 10 R q_{13}$	$q_{11} 1-$	$q_{12} 11 L q_{14}$
$q_{13} 00 R q_{10}$	$q_{14} 00 L q_{15}$	$q_{15} 00 R q_{16}$	$q_{16} 00 R q_{15}$
$q_{13} 11 R q_{24}$	$q_{14} 11 L q_{11}$	$q_{15} 11 R q_{17}$	$q_{16} 11 R q_{10}$
$q_{17} 00 R q_{16}$	$q_{18} 00 R q_{19}$	$q_{19} 01 L q_3$	$q_{20} 01 R q_{18}$
$q_{17} 11 R q_{21}$	$q_{18} 11 R q_{20}$	$q_{19} 11 R q_{18}$	$q_{20} 10 R q_{18}$
$q_{21} 00 R q_{22}$	$q_{22} 01 L q_{10}$	$q_{23} 01 R q_{21}$	$q_{24} 00 R q_{13}$
$q_{21} 11 R q_{23}$	$q_{22} 11 R q_{21}$	$q_{23} 10 R q_{21}$	$q_{24} 10 L q_3$

图 4-1 UTM（24, 2）的规则集

因为 UTM（24,2）被设计为纸带两端无限的图灵机，而本系统的仿真图灵机的纸带仅右端无限，因此需要增添几条规则:

start-0-0-R-t1；start-1-1-R-t1；

t1-0-0-R-t1；t1-1-1-R-t1；

t1-:-1-R-t2；t2-:-0- -q1。

这几条规则用于将图灵机的读写头从纸带最左端移动到 P 部分与 S 部分的分割处——“::”，并将其更改为 10，即 P0，然后读写头指向 S 部分的左侧，即 P 部分的最后一个字符，且图灵机此时的状态为 q1。此外，为图灵机增添正常结束状态 end，将规则$q_{11}1-$更改为 q11-1-1- -end。

下面说明该图灵机的程序输入格式要求，以标签系统 T: $a_1 \to a_2 a_1 a_3, a_2 \to a_1, a_3 \to STOP$为例，其中

$$N_1 = 1,$$

$$N_2 = N_1 + m_1 + 2 = 1 + 3 + 2 = 6,$$

$$N_3 = N_2 + m_2 + 2 = 6 + 1 + 2 = 9。$$

因此

$$P_0 = 10,$$

$$P_1 = 1010(00)^9 10(00)^1 10(00)^6 10,$$

$$P_2 = 1010(00)^1 10,$$

$$P_3 = 1110。$$

对于将由该标签系统转换的初始字符串$a_2 a_1 a_1$，其编码为

$$S = (01)^6 11(01)^1 11(01)^1。$$

经标签系统转换后，字符串变为a_3a_1，其编码为

$$S' = (01)^9 10 (01)^1$$。

该例子中，图灵机纸带上的初始字符串为

$P_3P_2P_1P_0S$=11101010001010100000000000000000000100010000000000000010::01010101010111011101。

预设模式下另一个可以选择的算法是二元加法[9]。该算法允许的字符集为{0,1}，用字符序列1^n表示正整数 n，用单个字符 0 分隔两个加数，其余位置的 0 均视为无效字符。对于正整数 x 与正整数 y 相加，其输入的字符序列为1^x01^y，运行结束后的结果字符序列为$1^{x+y}0$。该算法的规则集为：

start-0-0-R-start；start-1-1-R-q1；

q1-0-1-R-q2；q1-1-1-R-q1；

q2-0-0-L-q3；q2-1-1-R-q2；q2-空-0-L-q3；

q3-1-0- -end。

4.3.2 逻辑控制

图灵机从初始状态出发，读取纸带上的字符，然后根据规则表上的规则，进行一系列动作，并将图灵机转换到一个新的状态，这一整个过程称为图灵机的逻辑控制。在本系统中，这一个过程主要在 TuringMachine 脚本的 Update 函数中完成。

在该函数中，首先判断图灵机是否处于运行状态。若非，则执行 TuringMachine.Init()并退出；若是，则判断当前帧是否为特定帧，这一判断的过程及作用在 3.2 节已经作了说明。若为特定帧，则进入逻辑控制的部分。

首先，应当介绍两个变量——Queue<Movement>类型的 movements 和 string 类型的 nextState。movements 存储接下来图灵机将要执行的动作，每次发生动作前从中出队一个动作即可，该设计是因为本系统的仿真图灵机每次状态转换之间允许多个动作，而每个特定帧图灵机只会执行一个动作，所以需要把每次获取的转换项中的多个动作存储起来，以待使用。nextState 即图灵机将要转换到的下一个状态，初始值为 null，当动作队列为空时，代表本次转换的所有动作完成，图灵机状态变为 nextState。

下面介绍仿真图灵机的逻辑控制的完整流程。

首先判断动作队列是否为空，若为空，再判断 nextState 是否为 null，若不是 null，则调用 register.updateState(nextState)更新图灵机状态。因为动作队列为空，此时需要根据当前状态和当前字符从规则表中获取转换项。调用 register.getState()和 header.getCurrentChar()获取当前状态和当前字符，调用 table.convert(currentState, currentChar)获取转换项。此时要注意获取到的转换项是否为空，若为空，则代表在规则表中找不到相应转换项，也就是说图灵机陷入了停机状态。若将进入停机状态，则将 nextState 置 null，isRunning 置 false，并根据当前状态弹出相应信息窗口提示用户。若转换项不为空，则将转换项中动作添加到动作队列中，并更新 nextState。上述步骤均只在动作队列为空时执行，执行完毕后，除非停机，否则动作队列必定不为空。随后调用 movements.Dequeue()从动作队列中出队一个动作，根据动作的类型，执行相应动作。Movement 类型的结构在 3.2 节有所介绍，判断 movement.command。若为 L，调用 header.leftMove()；若为 R，调用 header.rightMove()；若为 E，调用 header.erase()；若为 P，调用 header.write(movement.c)。

在此贴出代码如下：

```
if (isRunning)
{
    if (frameNum % speedFactor == 0)
    {
        if (movements.Count == 0)
        {
            //更新状态
            if (nextState != null) register.updateState(nextState);
            //获取当前状态与指向字符
            string currentState = register.getState();
            char currentChar = header.getCurrentChar();
            //获取转移项
            Transfer transferLine = table.convert(currentState, currentChar);
            if (transferLine == null)
            {
                nextState = null;
```

```
                isRunning = false;
                Debug.Log("停机");
                return;
            }
            //添加动作到动作队列
            foreach (var MM in transferLine.movements)
            {
                movements.Enqueue(MM);
            }
            //更新下一个状态
            nextState = transferLine.nextState;
        }
        //获取下一个动作并运行
        Movement movement = movements.Dequeue();
        switch (movement.command)
        {
            case Transfer.Command.L:
                header.leftMove();
                break;
            case Transfer.Command.R:
                header.rightMove();
                break;
            case Transfer.Command.E:
                header.erase();
                break;
            case Transfer.Command.P:
                header.write(movement.c);
                break;
        }
    }}
```

第 5 章 部署与测试

5.1 部署

将用 Unity 制作的项目部署到服务器大致可以分为三步：

1. Unity 导出 WebGL 项目；
2. 服务器的设置；
3. 将 WebGL 项目上传到服务器上。

Unity 导出 WebGL 项目需要下载相应模块，然后选择 Build Settings-Build 即可。导出的项目包含以下几个部分：

1. index.html 文件，浏览器可通过导航该文件来加载内容；
2. 包含构建徽标、加载进度条和其他模板资源的 TemplateData 文件夹；
3. 包含生成的构建输出文件的 Build 文件夹。

导出项目后我们需要一个服务器来处理浏览器用户的请求并将项目传递给用户，然后浏览器解析并展示系统。在此本系统选择了 Apache HTTP 服务器并将其运行在装载有 Ubuntu 系统的云服务器上。选择 Apache 服务器的原因是其开源且具有较好的跨平台性和安全性。

第三步，我们将项目文件上传到云服务器上。对于本系统这样只具有一个站点的情况，我们可以将整个文件拷贝到/var/www/html 文件夹下，该文件夹是 Apache 服务器默认的主站点访问路径。

至此，我们完成了本系统的部署工作。之后修改/etc/apache2/ports.conf 文件以更改服务器监听端口为 2002，然后使用 systemctl start apache2 命令启动服务器。此时用户就可以通过在浏览器地址栏输入 http://服务器 IP 地址:2002 来访问本系统了。

5.2 测试

此处进行的测试为系统测试，此外，因为本系统相对简单且为个人开发完成，

我们省略了一些不必要的测试步骤以减少工作量。系统测试将在最终环境下进行，即将系统部署到云服务器，然后通过浏览器访问该系统，并验证其各个功能是否正常运行且符合要求。以下为测试步骤。

在浏览器输入系统所在地址，经短暂的 Unity 加载界面后，系统界面出现，如图 5-1 所示。

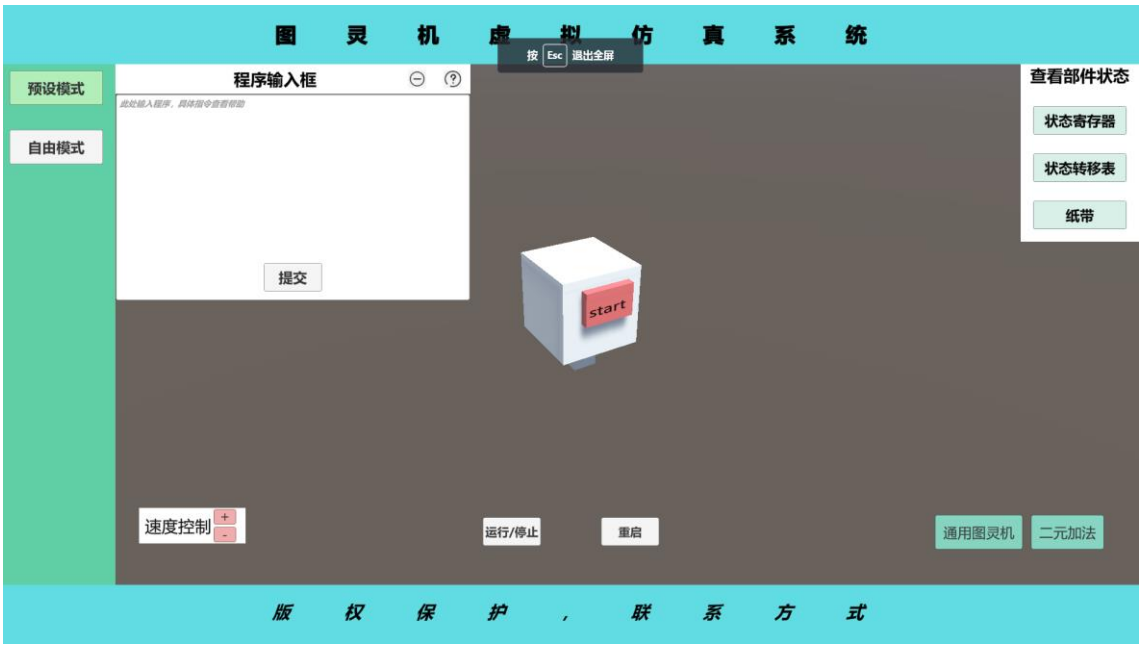

图 5-1 系统界面

首先测试预设模式通用图灵机算法的表现。测试流程如下：

1. 初始系统默认进入预设模式，点击右下角算法选择板块的通用图灵机按钮；
2. 在程序输入框输入以“::”分隔的，左侧为 2 标签系统的规则，右侧为 2 标签系统的输入的字符序列 11101010001010100000000000000000000100010000000000000010::01010101010111011101，点击提交按钮，图灵机模型下方出现纸带，纸带上字符与输入相同；
3. 点击运行/停止按钮，图灵机由停止状态进入运行状态，按照规则表（状态转移表）中的规则进行动作；
4. 运行一段时间后，系统弹出停机的信息提示窗口，表明运行结束；
5. 查看运行结果，点击查看部件状态栏下纸带按钮，弹出纸带状态窗口，显示结果与预期相符。

然后测试二元加法算法的表现，测试流程与上述类似。程序输入框输入 1101110，意为 2+3，纸带上结果为 1111100，即 5，与预期相符。

同时借由该算法测试各功能按钮的正确性。在运行过程中点击“运行/停止”按钮，图灵机立即停止运行。点击“重启”按钮，纸带和寄存器均回到初始状态，注意此时纸带上内容并没有变化，点击程序输入框的“提交”按钮，才能使纸带上内容回到初始状态。图灵机运行时，点击速度控制“+”和“-”按钮，图灵机运行速度均能做出相应变化。分别点击查看部件状态部分的三个按钮，所对应窗口均能正常显示。具体运行画面见图 5-2。

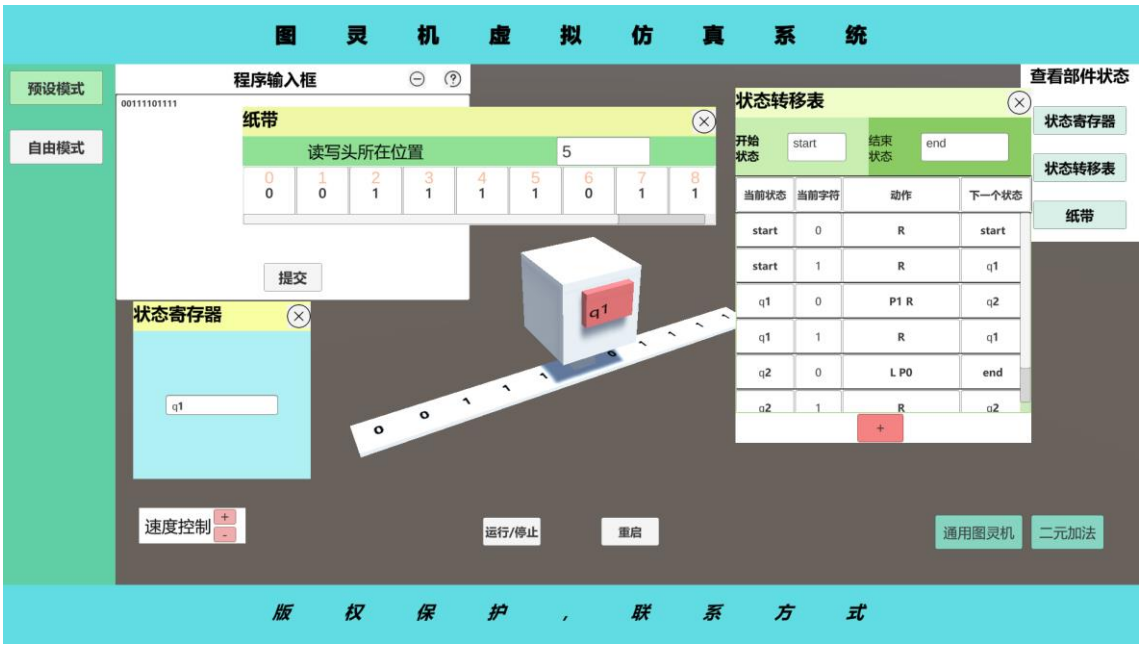

图 5-2 运行时界面

接下来测试自由模式下该系统的运行状态。首先点击左侧的自由模式按钮使系统进入自由模式，此时图灵机各个部件状态都被清空，查看各部件状态窗口可观察到这一点。通过在状态转移表状态窗口输入规则自定义图灵机，在此以将输入的所有字符“0”改为“1”这样的图灵机作为测试样例，其规则表为：

q0-0-P1-R-q0；q0-1- -R-q0；q0-空- - -end。

在程序输入框输入字符序列：001001110，图灵机停机时序列中所有 0 被改写为 1。接下来以该样例为依据，测试自由模式下图灵机未停机时更改各部件状态后，图灵机对此做出的响应。

重启图灵机，重新提交字符序列，点击运行待图灵机运行几步后点击停止。在状态寄存器状态窗口修改当前图灵机状态为“end”，之后点击运行，图灵机立即停机，测试成功。

重新初始化图灵机，运行后停止，在状态转移表状态窗口修改规则表为：

q0-0- -R-q0；q0-1-P0-R-q0；q0-空- - -end。

即将后续纸带上所有的“1”改为字符“0”。同时在纸带状态窗口修改纸带内容，确保后续纸带中存在字符“1”。完成如上操作后点击运行，纸带上后续内容如预期全被改为 0。

至此，该系统主要功能测试完毕。

第 6 章 结论

本研究设计并实现了一个面向教学的图灵机仿真系统。本系统的目的是帮助新手学习者在学习图灵机和其他计算机科学核心概念时，提供一个主动学习的环境。通过这一仿真系统，学生可以亲自操作图灵机，观察其运行过程，从而深入理解其原理。同时，该系统还可以作为教师在课堂上的辅助教学工具，帮助学生更好地掌握计算机科学相关的基础知识。

本系统具有两个独特的模式：预设模式和自由模式。在预设模式下，系统通过模拟 2 标签系统来模拟通用图灵机的运行。用户可以在这一模式下观察图灵机的运行过程，了解图灵机的一般运行原理，并研究通用图灵机的实现方式。这种模式特别适合初学者，可以帮助他们逐步建立起对图灵机的基本理解。而在自由模式下，用户则拥有对仿真图灵机的绝对控制权。他们可以根据自己的需求和兴趣，自由设计算法并进行计算理论研究。这一模式为用户提供了更广阔的探索空间，有助于激发他们的创造力和创新精神。

综上所述，面向教学的图灵机仿真系统是一种非常有效的教学工具。它不仅可以帮助学生更好地理解和掌握计算机科学的核心概念和技术，还可以为他们提供一个自由探索、创新实践的平台。通过这一系统，学生可以更加深入地了解图灵机的原理和应用，为未来的学习和研究打下坚实的基础。

尽管本系统已经具备了一定的功能和应用价值，但仍然存在许多不足和可改进之处。在用户界面方面，整体设计显得相对简陋，缺乏直观性和易用性，这可能会影响用户的使用体验和操作效率。同时，系统提供的指引和说明也相对较少，可能导致用户在使用过程中遇到困惑和疑问。

此外，在预设模式方面，目前系统主要提供的是通用图灵机，但对于某些复杂算法的研究和应用，可能需要更加专业的图灵机预设。因此，未来可以考虑增加更多专用图灵机的预设模式，以满足不同用户的需求和研究方向。这将有助于提升系统的功能和性能，进一步推动相关领域的研究和应用发展。

参考文献

[1] Alan Mathison Turing. On Computable Numbers, With an Application to the Entscheidungsproblem[J]. Proceedings of the London Mathematical Society, 1936,Series 2,Volume 42:230-265+544-546.

[2] 陈凯.怎样动手“造”一台图灵机[J].中国信息技术教育,2018,(09):27-29+39.

[3] Mike Davey.A Turing Machine In The Classic Style[EB/OL].[2024-04-30]. https://aturingmachine.com/index.php.

[4] 郭惠芳.用 sendmail 仿真一个图灵机[J].计算机应用与软件,2008,(01):170-172.

[5] 郭楠,丛欣宇,吕雅静,等.基于 Unity 的图灵机虚拟仿真系统[P].辽宁省:CN202210402665.X,2022-07-29.

[6] C.E.Shannon.A universal Turing machine with two internal states[J].Automata Studies,Annals of Mathematics studies,1956,no.34:157-165.

[7] M.L.Minsky.Size and structure of universal Turing machines using tag systems[J]. Journal of Symbolic Logic,1966,Volume 31 (4):655-655.

[8] Yurii Rogozhin.Small universal Turing machines[J].Theoretical Computer Science,1996,Volume 168, Issue 2:215-240.

[9] 王强.四则运算图灵机的构造[J].内蒙古师范大学学报(自然科学汉文版),2004,(03):275-277.

[10]蒋宗礼,姜守旭.形式语言与自动机理论[M].2 版.北京:清华大学出版社,2007:282-283.

致　　谢

时光荏苒，大学四年眨眼而过。在此，我衷心感谢这个过程中所有给予我支持和帮助的人。感谢家人和朋友的陪伴与支持，感谢各科任课教师的悉心教导，感谢指导教师郭东伟教授对我毕设工作的帮助。