Periotris.js：类俄罗斯方块化学元素周期表学习系统

鲍溶

宁波效实中学 D2303

摘要

在本文中，作者描述了一种通过信息技术辅助用户学习、记忆化学元素周期表的完整交互式学习系统，其采用类似“俄罗斯方块”游戏的规则，通过鼓励用户将上写有元素符号的四联牌放置于正确的位置强化用户对元素周期表结构与内容的掌握程度。该系统的核心是一套快速生成算法，其能够针对任意符合要求的图形生成一系列能够恰好密铺该图形的多联牌，以及一个能让上述多联牌无阻挡地下落、拼贴为原始图形的下落次序。上述算法具有通用性，对包括标准元素周期表图形在内的多种图形适用。基于此算法开发的上述交互式学习系统在多种环境中均能达到较高的性能。

关键字：元素周期表 俄罗斯方块 图案密铺 精确覆盖问题 用户交互

Periotris.js：类俄罗斯方块化学元素周期表学习系统

化学元素周期表作为化学学科的标志之一 (Wikipedia, 2022)，其重要性不言而喻。Emsley (1989) 在他的著作中认为，“只要人们还在研究化学，元素周期表就不会消失。”元素周期表可以清晰地展示元素性质的周期性变化，从而引导人们外推已有结论，发现新的元素。也正由于元素周期律的多样性，从Mendeleev在1869年发明第一份现代意义上的元素周期表开始，在随后的150多年中，人们陆续发明了大约1000种元素周期表 (Scerri, 2020)。尽管周期表的样式多种多样，学习元素周期表中元素的排布仍然是一项艰巨的任务，学生们在记忆元素时仍感觉“繁琐无趣” (林, 2021)，当前较为传统的教学方法亟需改进。

黎卓熹等人 (2020) 分析了多项2009年至2019年的部分文献，指出国外教授元素周期表相关知识时通常采用构建模型、探究学习、游戏学习、合作学习等方式。近期的报道，比如发明化学卡牌游戏 (Martí-Centelles & Rubio-Magnieto, 2014)、将化学与乒乓球运动结合 (Lee, et al., 2016)、将化学知识应用于密室逃脱关卡的设计 (Dietrich, 2018) 等等方案，为学生的元素周期表学习提供了更多乐趣，但上述方案都未能考虑到COVID-19对线下集体教学活动带来的影响。

“俄罗斯方块”（Tetris1）是一种经典解谜游戏，玩家需要控制四联牌（tetromino2）下落至合适的位置，使四联牌堆叠高度尽可能晚地超过游戏区域高度限制。若将该游戏中“行填满时便消除得分”的机制删除，同时精心设计不同四联牌下落的次序，便可以得到一种图案拼贴游戏，其游戏目标变化为“使用四联牌堆叠出某种图形”。若使用元素周期表的图案设计下落次序，就能设计出一种通过俄罗斯方块的机制实现元素周期表教学的系统。

该设计思路的难点有二：一是如何将一张标准元素周期表切分为相应的一系列四联牌；二是如何找出一个恰当的下落次序，使四联牌能以该顺序不受阻挡地落于正确的位置。当上述两点均被恰当实现之后，仅需将切分后的四联牌进行平移和随机旋转，便可以实现俄罗斯方块游戏中四联牌从游戏区域顶端下落的效果。用户仅需对四联牌进行平移、旋转两种操作便可以将其放置于正确位置。

基于该思路，作者设计了如[图1](#图1)所示的系统。为了达到功能分离的目的，作者使用了*模型——视图——视图模型*（Model-View-ViewModel） (Gossman, 2008) 的设计模式，见[图2](#图2)。视图包括最终绘制于用户界面上的交互元素，如按钮、文字等，基于已有的基础设施和框架构建，如Gatsby框架 (Gatsby, Inc., 2022) 和 MUI可重用网页组件库 (Material-UI SAS, 2022)。视图模型接收从视图传入的用户交互信息，并对模型进行相应修改，同时将修改后的状态通过事件形式实时刷新至视图。该刷新操作由状态管理器MobX (MobX contributors, 2022)完成。模型负责所有抽象逻辑的管理，包括所有交互逻辑和四联牌生成。为了更好地分担工作并防止绘图线程被阻塞，模型管理了一个Web Worker，并将生成算法运行于Web Worker之中。该系统在实际应用中有良好的表现，在保持高性能的同时，能够起到提升元素周期表学习效率的作用。

## 1. 生成算法的设计

### 1.1. 多联牌切分

多联牌切分算法是本系统生成算法的第一部分，其本质是尝试使用若干多联牌完全密铺给定的模板。Knuth (2000) 认为深度优先搜索是解决此种覆盖问题的“唯一合理解法”，因此，作者也将使用深度优先搜索进行多联牌切分。在以下讨论中，我们取任意的*K-联牌* （其中）作为多联牌的一种不失普遍性的样例。

我们定义*模板空间* 是一个 的0/1矩阵。其中，值为1的元素代表“该方格需要被密铺”，反之亦然。因此，*模板* 则是由以下公式定义的点集：

同时，定义K-联牌 如下：

其中 指示了该K-联牌的*类型*； 是一个*K*元集合，包含了其所覆盖的所有点在矩阵中的行/列坐标。K-联牌的类型数量指的是K个方格边边相连的所有可能情况中，无法通过平移、旋转两种操作重合的情况数量，其中当 时的各种类型及其在本文中的命名见[图3](#图3)。那么，多联牌切分算法的目标，是找到一个由 构成的集合 ，使其满足以下条件：

此时 便是该算法的输出。该算法接受三个输入：模板空间 ，K-联牌 的值，以及希望使用的K-联牌的类型集合 。实现该算法的、以Cormen等人 (2009, pp. 16-22) 描述的格式3写出的伪代码如下：

Tetrominify(, , )

let be the template generated from template space

**if**

**error** “unable to tile”

let be an empty stack

let be an empty stack

**while** True

**if**

**return**

**if** we need to trace back, i.e. all attempts in this subtree have failed

**if**

**error** “unable to tile”

pop an element from

pop an element from into

**else**

let be a clone of

let be the coordinate of the left-most upper-most 0 in

**repeat**

generate a polyomino of category with random direction, whose left-most upper-most block is at

**if** and

push to

push all untested categories in to

inform that we don’t need to trace back

**break**

**until** all categories has been tested

该算法首先根据一些简单规则判断是否有可能对给定模板空间使用K-联牌进行密铺。这些规则是启发性的，不保证能够完全覆盖所有无法密铺的情况。之后，算法将从模板空间的最左上方开始生成用户指定类型的K-联牌，并且像所有回溯算法一样，维护了一个包含在每个分支未尝试选项的栈。当某个K-联牌既没有覆盖任何被模板空间禁止的方格，同时也没有与已放置的K-联牌产生冲突时，便可以被压入结果栈。如果所有当前方格不能被任何满足上述条件的K-联牌覆盖，算法便回溯一步，放弃该节点并从上一个节点的另一分支继续搜索。若当前节点为根节点但依然要求回溯，那么对于给定的参数 不存在任何密铺方案，算法将报告一个错误。

该在最坏情况下，运行该算法需要为 个K-联牌分配的辅助空间和，则该算法的空间复杂度为。同样地，由于该算法最坏情况下必须遍历所有的路径，因此该算法的时间复杂度为。在实际应用中，该算法的生成速率属于秒级。

该算法可以简单地衍生为一种随机算法，仅需将“选择一种类型 ”的选择方式从顺序选择改为任意选择。请注意，尽管不同方向的随机性较大，应该尽量保持类型的局部性，也就是说将同一类型的可能性时间上尽可能近地考虑，以保证更佳的最终性能。

### 1.2. 下落顺序生成

仅能够切分多联牌并不能完成本系统对生成算法的要求。若仅依据多联牌切分算法的随机性确定下落顺序，很容易出现*不合法的下落顺序*，即在俄罗斯方块的游戏规则下，使用该下落顺序依次生成多联牌，不能将其拼贴为原始的模板。考虑[图4](#图4)中的一个由三块四联牌所密铺的模板。在该模板总数为 的所有下落顺序中，仅有方形四联牌第一个下落的2种是*合法*的，而其他所有下落顺序均不合法，因为在这些顺序中，方形四联牌会被其他两个四联牌突出的部分所阻挡而无法到达还原原始模板所要求的正确位置。

为了解决这种问题，我们需要对所有生成的多联牌进行*拓扑排序*以保证多联牌之间正确的下落顺序。拓扑排序是一种在*有向无环图*（DAG，**d**irected **a**cyclic **g**raph）上的算法，其可以为所有图上的*顶点*指定一个顺序，在该顺序下的所有*边*都指向同一个方向。拓扑排序在解决对象依赖问题方面有重要作用，比如其可以用于解决早晨我们每个人都要纠结的穿衣顺序问题 (Cormen, Leiserson, Rivest, & Stein, 2009)。

下落顺序生成算法的第一步是对多联牌集合生成一个DAG以表示各个多联牌之间的依赖关系。根据俄罗斯方块规则，我们对K-联牌 定义*依赖集合* ：

这样，我们能够保证 依赖集合的元素都是在 下方且与它有邻边的K-联牌，这些K-联牌都必须先于 下落。

基于以上讨论，我们可以写出生成依赖集合的算法。该算法有三个参数输入：K-联牌 、K-联牌的 值以及所有经过多联牌切分所得到的K-联牌集合 ，有一个输出：依赖集合 。

Get-Dependencies(, , )

let be

let be an empty set

**for** **to**

let be

**for** **to**

**if** contains

push into

**return**

类似地，我们还可以定义被依赖集合 ：

当获得了每个K-联牌的 和 后，我们便可以使用Kahn (1962) 描述的算法对其进行快速拓扑排序，代码如下：

Topo-Sort(, )

for each polyomino in , associate the result of Get-Dependencies(, , ) and Get-Dependents(, , ) to as and

let be the starting node set defined by

let be an empty stack

run Kahn’s algorithm on and store the result in

**if**

**error** “one or more cycles are found”

**return**

注意到该算法是一个随机算法，因此其可能对于同一K-联牌的集合产生不同的下落顺序。这是有意的，因为可以借此增加生成算法的随机性，避免应用层面上游戏性的缺失。

### 1.3. 合法下落顺序存在性

上述下落顺序生成算法在依赖图为一个DAG时可以正确生效，当依赖图包含环时会报告错误。遗憾的是，我们无法保证多联牌切分算法在任意 下都能够生成没有循环依赖的密铺方式，即，在 时，不能保证一定存在合法的下落顺序。该条件还可以更加紧确。考虑如[图5](#图5)所示的情形。在图中所示的 的一个切分中存在了一个二元依赖环，该依赖环由6个方格组成，两个五联牌分别贡献了1个方格和5个方格。我们将该模式推广，可以发现，无论是在水平方向上增加其中上下两个方格的距离——也就是改变其中一个多联牌的形状，还是增加竖直方向上被包裹的多联牌个数——也就是增加依赖环的元数，都会导致一个多联牌所覆盖方格数的增加，也就是 值的增加。因此，我们可以确认，当 时，可能会出现循环依赖；当 时，由于多联牌覆盖方格数小于产生循环依赖的最小数量，所以恰好不会产生循环依赖。

有了上述结论，我们可以肯定，在本系统的构建中，若仿照俄罗斯方块游戏的设计选择 ，恰好可以避免循环依赖的出现，也就能保证对于某个切分结果，定有至少一种合法的下落顺序。

## 2. 系统的软件实现

该系统的实现可以自然地分解为两部分：呈现与交互逻辑部分和核心算法生成部分。为了最大程度上使两部分解耦合，我们采取先独立再综合的方式，使用简单易用的.NET框架分别构建了两个实验性组件：简单的俄罗斯方块游戏“SimpleTetris”和四联牌切分工具“Tetriminify”，后者的运行界面见[图6](#图6)。在确认了两个组件均能正常工作后，我们使用Gatsby.js作为框架，使用TypeScript语言完成了本文描述的系统“Periotris.js”的构建，并对其进行了一系列调整和优化。

*模型*（model）的主体是一个名为GameModel的*类*（class）。该类继承了事件产生器EventEmitter类，可以发送用户指定的*事件*（event）。模型存储了所有的抽象数据，包括已落下的、已激活的4和等待激活的四联牌。[图7](#图7)展示了模型与外界的信息交换的三种途径：借助*属性*（props，**prop**ertie**s**）读取，数据单向外传；借助方法调用，数据单向内传；借助事件，数据单向外传。外界通过读取属性和监听事件获取模型当前的状态，如当前所有可见方格的状态、已激活的四联牌位置，等等；外界通过调用方法安全地操纵数据，如移动、旋转当前激活的四联牌，同时无需担心损坏数据，比如在还未完成四联牌切分时用户试图旋转四联牌的操作不会有任何效果。模型能够自行保证数据始终处于可预测的状态。同时，即使删除所有除模型部分之外的代码，模型也能够正常工作。模型还具有*时序无关*的特点：由外部定时器决定的对系统状态更新的速率不会影响模型的工作。模型的各项数据接口被设计为可以任意速率正确访问。

*视图模型*（view model）同样是一个类，名为GameViewModel。视图模型管理了一个模型的*实例*（instance），负责监听、解读模型的各种状态变化，并恰当地操作模型。视图模型维护了两个定时器，使其按照预定的时间间隔发送事件，分别用于游戏进度的更新和游戏时长的更新。视图模型负责将每次模型状态的变更转换为自身多个属性的变化，同时通知状态管理器MobX (MobX contributors, 2022) 提交给视图进行刷新。这种转换的一个例子，是将模型中每个方格所代表的元素的内部表示转换为用户可读的元素符号及颜色。状态的刷新是自动的，得益于MobX提供的*观察者模式*（Observer pattern）的API。视图模型同时监听用户输入，并调用适当的方法对模型进行操作。比如，当用户按下“W”键时，视图模型将调用模型暴露的旋转已激活四联牌的方法。当用户划动屏幕时，视图模型将通过手指划动的方向将四联牌向对应的方向移动。视图模型的作用是“管道”，连接视图和模型。

*视图*（view）是呈现给用户的部分。视图由各类*组件*（component）构成。这些组件大部分是可以自我包含（self-contained）的可重用部分。我们使用React (Meta Platforms, Inc., 2022) 进行视图的构建。所有视图元素都处于一个全局的*上下文*（context）中，该上下文包括了一个同样是全局的视图模型实例。需要根据游戏进度做出响应的组件在上下文中注册为一个*观察者*（observer），并在视图模型更新时同步刷新。典型的例子包括用于呈现游戏主体区域的*方块网格*（block grid），其承担了渲染所有方格及格线（详见[图1](#图1)），包括已落下和已激活的四联牌。用户通过视图与系统进行交互，包括但不限于按下按键、划动手指和点击屏幕上的内容。

*基础设施*（infrastructure）包括支持本系统运作的所有其他工具与环境。本系统主要使用Gatsby.js工具链 (Gatsby, Inc., 2022) 构建，其内部使用了包括Webpack (webpack contributors, 2022) 在内的多项技术，在这篇文章内涉及所有这些方面是不现实的。为了加强用户体验，防止CPU密集型的四联牌切分*阻塞*（block）绘图线程，我们使用了Web Worker (WHATWG, 2022)。这是一种在浏览器内部实现并行编程的方法。通过将生成算法的实现移至绘图线程之外，我们极大地提升了系统的响应速度和用户体验。同时，通过支持*渐进式互联网应用*（PWA，**P**rogressive **W**eb **A**pps） (Google, 2022) ，我们让本系统能够以类似原生应用的形式安装至用户的终端中，提供了离线环境下的较高质量体验。

一个大致的游戏流程可以描述如下：用户点击“开始”按钮，此时Web Worker中的生成算法开始运行并生成一系列四联牌和相应的一种下落顺序，并将其返回给模型。模型收到该结果后，将所有四联牌移至初始位置——屏幕上方中间位置，并作随机旋转。此时游戏状态变更为“游戏中”，视图模型监听到此次改变，将当前已激活四联牌的信息转化为渲染组件信息并刷新至自身的一个列表中。视图通过状态管理器将更新后的渲染信息绘制到屏幕上。此时用户便能看到屏幕上出现了第一个四联牌。之后用户做出相应的动作，通过视图模型反应至模型，模型作出相应的变更，这些变更最终又被刷新至屏幕，以此循环往复直到用户完成拼贴，或者拼贴失败，在任意一种情况下，游戏结束。

## 3. 结果

经过多个版本的迭代，Periotris.js已经具有了相当的实用价值。从生成算法的多次优化的效果来看，目前Periotris.js 2.0.2具有秒级的生成速率，在包括桌面和移动端的多种浏览器平台上均有出色的表现，详细数据参见[表1](#表1)。

Periotris.js还得到了多位试用者的好评。[图8](#图8)展示了一位试用者体验Periotris.js的情形，注意其将系统设置为“无颜色渲染”模式以提高游戏的难度。近期，Periotris.js还在宁波效实中学初三学生冬令营的化学课堂上得到了应用，使用者对此的评价是“速度很快”。

## 4. 总结与发展方向

Periotris.js是一套能够辅助化学元素周期表教学与记忆的完整系统。通过具体化精心设计的生成算法，该系统能够将元素周期表作为模板切分为若干四联牌，并以俄罗斯方块游戏的形式引导用户重新将下落的四联牌通过平移、旋转还原为原始的元素周期表，以达到加强用户对元素周期表结构与各元素相对位置的记忆和理解的作用。该系统具有较佳的性能，同时也在用户交互方面受到好评。

未来的工作可以在以下方面展开：一是增加模板空间数量，比如加入仅包含短周期的模板空间，以及包含完整镧系和锕系的周期表。由于该系统底层算法的通用性，作者认为这点应当不难实现；二是增加更多的个性化选项，比如自定义色彩主题等，鉴于本系统采用视图与模型分离的架构设计，我们也应当较容易实现多样的自定义功能。

## 5. 源代码可见性

Periotris.js的源代码可于<https://github.com/CSharperMantle/periotrisjs/> 获取，采用GNU General Public License Version 3 or Later (Free Software Foundation, Inc., 2007) 授权。SimpleTetris和Tetriminify的源代码可于 <https://github.com/CSharperMantle/CmTetris/> 获取，同样采用GNU GPL v3 or Later授权。也可以向作者发送电子邮件获取相关源代码。

两个Periotris.js的实例分别托管于GitHub Pages和Vercel。可以在 <https://csharpermantle.github.io/periotrisjs/> 和 <https://periotrisjs.vercel.app/> 分别找到上述两个实例。

## 6. 致谢

*“As you set off on your journey once again, you must remember that the journey itself has meaning.”* (Barbatos, 2020)

我们希望向宁波效实中学化学组徐珑迪老师表示诚挚的感谢。他对元素周期表的导论性课程激发了作者对该选题的无限兴趣。在刚刚过去的2021年中，他提出了许多建设性意见，并且在本项目断断续续的、长达一年的开发周期中，始终鼓励作者牢记出发的目的。

引用

Barbatos, V. (2020). Prologue: Act III - Song of the Dragon and Freedom, Ending Note. *Genshin Impact*. Shanghai: miHoYo, Inc.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms, Third Edition.* The MIT Press.

Dietrich, N. (2018). Escape Classroom: The Leblanc Process—An Educational “Escape Game”. *Journal of Chemical Education, 95*(6), pp. 996–999. doi:10.1021/acs.jchemed.7b00690

Emsley, J. (1989). *The Elements.* Oxford: Clarendon Press. doi:10.1002/bbpc.19900940223

Free Software Foundation, Inc. (2007). *The GNU General Public License v3.0.* Retrieved February 14, 2022, from GNU Project - Free Software Foundation: http://www.gnu.org/licenses/gpl-3.0.html

Gatsby, Inc. (2022). *The Fastest Frontend Framework for Headless CMS's*. Retrieved February 13, 2022, from Gatsby: https://www.gatsbyjs.com/

Google. (2022). *Progressive Web Apps.* Retrieved February 14, 2022, from web.dev: https://web.dev/progressive-web-apps/

Gossman, J. (2008, May 28). *PresentationModel and WPF.* Retrieved February 13, 2022, from Microsoft Docs: https://docs.microsoft.com/en-us/archive/blogs/johngossman/presentationmodel-and-wpf

Kahn, A. B. (1962). Topological sorting of large networks. *Communications of the ACM, 5*(11), pp. 558–562. doi:10.1145/368996.369025

Knuth, D. E. (2000). Dancing links. In J. Davies, B. Roscoe, & J. Woodcock, *Millennial Perspectives in Computer Science: Proceedings of the 1999 Oxford-Microsoft Symposium in Honour of Sir Tony Hoare (Cornerstones of Computing)* (pp. 187-214). Red Globe Press.

Lee, C.-H., Zhu, J. F., Lin, T.-L., Ni, C.-W., Hong, P. C., Huang, P.-H., . . . Ho, M.-L. (2016). Using a Table Tennis Game, “Elemental Knock-Out”, To Increase Students’ Familiarity with Chemical Elements, Symbols, and Atomic Numbers. *Journal of Chemical Education, 93*(10), pp. 1744–1748. doi:10.1021/acs.jchemed.6b00341

Martí-Centelles, V., & Rubio-Magnieto, J. (2014). ChemMend: A Card Game To Introduce and Explore the Periodic Table while Engaging Students’ Interest. *Journal of Chemical Education, 91*(6), pp. 868-871. doi:10.1021/ed300733w

Material-UI SAS. (2022). *MUI: The React component library you always wanted*. Retrieved February 13, 2022, from MUI.

Meta Platforms, Inc. (2022). *React – A JavaScript library for building user interfaces.* Retrieved February 14, 2022, from React: https://reactjs.org/

MobX contributors. (2022). *README*. Retrieved February 13, 2022, from MobX: https://mobx.js.org/README.html

Scerri, E. (2020). *The Periodic Table: Its Story And Its Significance.* New York, NY: Oxford University Press.

Tetris Holding. (2022). *About Tetris®*. Retrieved February 12, 2022, from Tetris | The addictive puzzle game that started it all!: https://tetris.com/about-us

webpack contributors. (2022). *webpack.* Retrieved February 14, 2022, from webpack: https://webpack.js.org/

WHATWG. (2022, February 11). *10 Web workers.* Retrieved February 14, 2022, from HTML Standard: https://html.spec.whatwg.org/multipage/workers.html

Wikipedia. (2022). *Periodic table.* Retrieved February 12, 2022, from Wikipedia: https://en.wikipedia.org/wiki/Periodic\_table

黎卓熹, 成燕琴, 魏洁书, 张镖, & 刘盈. (2020). 国外化学元素周期表教学方式的介绍与启示——基于2009-2019年Journal of Chemical Education的载文. 化学教育, 41(11), 页 107-113. doi:10.13884/j.1003-3807hxjy.2019110158

林郴. (2021). 对当下高中化学学科教学的评论. [私人访谈].

脚注

1 Tetris是Tetris Holding的商标。

2“Tetromino”一词由“domino”（多米诺骨牌）衍生而来，本意为“由四个小方块以边相连所构成的各种图形”。在俄罗斯方块游戏中，特将这些四联牌称为“tetrimino” (Tetris Holding, 2022)。本文将尽可能使用“四联牌”这一称呼，在描述通用算法时使用“tetromino”，在描述具体应用实现时使用“tetrimino”。

3 本文中伪代码将均使用Cormen等人所提供的格式书写。

4 “激活的”四联牌指用户可以操纵的四联牌。一般在某一特定时刻只能由不多于1个激活的四联牌存在。

附表

表1

*多平台下生成算法的性能统计*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Measure # | Laptop | Desktop | | Mobile | |
| Browser 1 | Browser 1 | Browser 2 | Browser 3 | Browser 4 |
| ​0 | 3303.00 | 223.00 | 40.30 | 679.00 | 4232.20 |
| ​1 | 4191.00 | 58.00 | 67.20 | 212.00 | 1833.40 |
| ​2 | 10.00 | 6.00 | 14.90 | 11.00 | 675.10 |
| ​3 | 618.00 | 704.00 | 124.40 | 97.00 | 105.60 |
| ​4 | 10.00 | 119.00 | 234.10 | 661.00 | 1946.90 |
| ​5 | 212.00 | 152.00 | 2273.30 | 88.00 | 392.10 |
| ​6 | 558.00 | 14.00 | 402.40 | 1258.00 | 4705.60 |
| ​7 | 10440.00 | 158.00 | 221.40 | 47.00 | 27.40 |
| ​8 | 262.00 | 457.00 | 935.50 | 2103.00 | 4545.50 |
| ​9 | 2295.00 | 4116.00 | 27.70 | 1305.00 | 264.50 |
| ​10 | 17.00 | 1394.00 | 47.80 | 1529.00 | 1428.30 |
| ​11 | 4855.00 | 2580.00 | 256.20 | 216.00 | 3152.70 |
| ​12 | 929.00 | 840.00 | 7.20 | 2219.00 | 1218.30 |
| ​13 | 2110.00 | 12.00 | 684.40 | 5600.00 | 1304.50 |
| ​14 | 12376.00 | 44.00 | 122.10 | 126.00 | 195.90 |
| ​15 | 374.00 | 131.00 | 18.70 | 4901.00 | 2352.90 |
| ​16 | 324.00 | 1785.00 | 2053.40 | 978.00 | 985.30 |
| ​17 | 85.00 | 21.00 | 18.40 | 1032.00 | 262.70 |
| ​18 | 33.00 | 175.00 | 3.40 | 954.00 | 6711.40 |
| ​19 | 1115.00 | 11.00 | 1426.60 | 473.00 | 52.29 |
| Average | 2205.85 | 650.00 | 448.97 | 1224.45 | 1819.63 |
| S. D. | 3477.56 | 1072.5 | 692.195 | 1531.16 | 1907.92 |

注释：测试用平台：Laptop：Windows 10 1511 Intel i3-4010U@1.70GHz；Desktop：Windows 10 21H1 Intel i3-4150@3.50GHz；Mobile：Android 11 (MIUI 12.5.3.0) Qualcomm Snapdragon 730G@2.2GHz。测试用运行环境：Browser 1：Firefox Developer Edition 98.0b4 (64-bit)；Browser 2：Microsoft Edge 98.0.1108.50 (64 bit) (V8 9.8.118.4)；Browser 3：Firefox Daylight 97.1.0；Browser 4：MIUI Browser 15.5。表格中所有数据均以毫秒（ms）为单位。

附图



图 1：Periotris.js 是一种通过信息技术辅助用户学习、记忆化学元素周期表的完整交互式学习系统，其以游戏“俄罗斯方块”的形式，交互地帮助用户学习、记忆元素周期表的结构与各元素的相对位置。

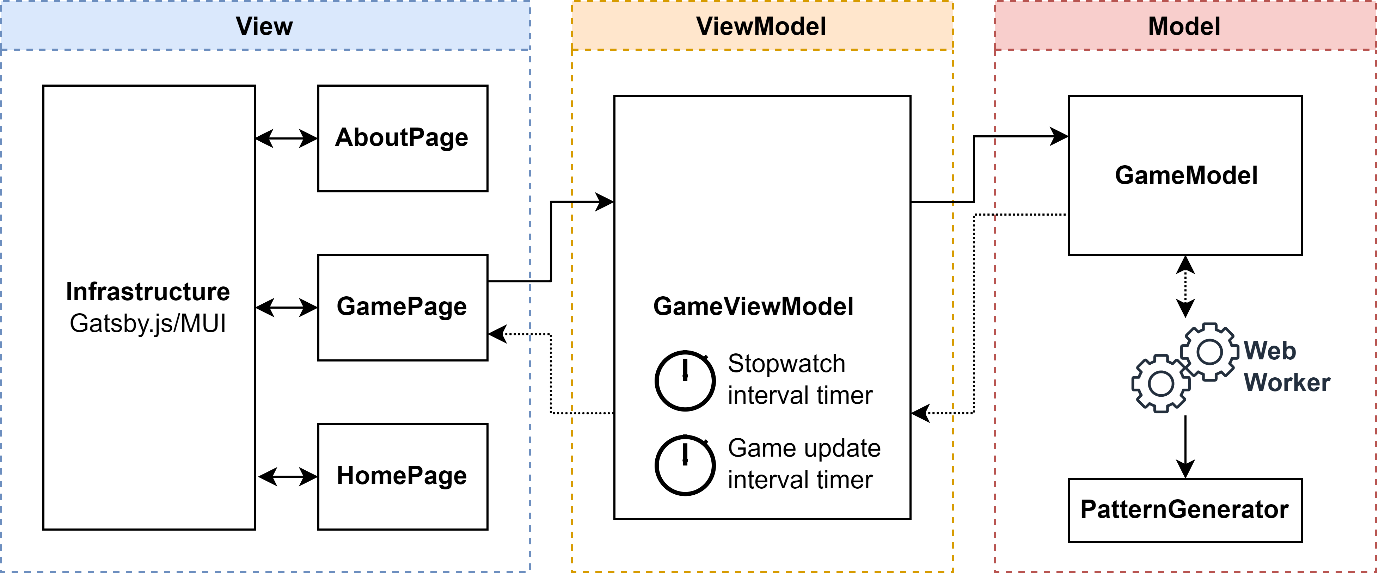


图 2：系统概览。图中实线代表方法调用，虚线代表事件发送。作者使用视图-视图模型-模型架构构建Periotris.js系统。视图基于已有的基础设施架构Gatsby.js和组件库MUI构建。视图模型负责处理用户输入并根据模型的变更对视图元素进行相应的更新。模型运行包括生成算法在内的多项业务逻辑，负责抽象数据处理，使用Web Worker实现并行化以防止绘图线程卡顿。

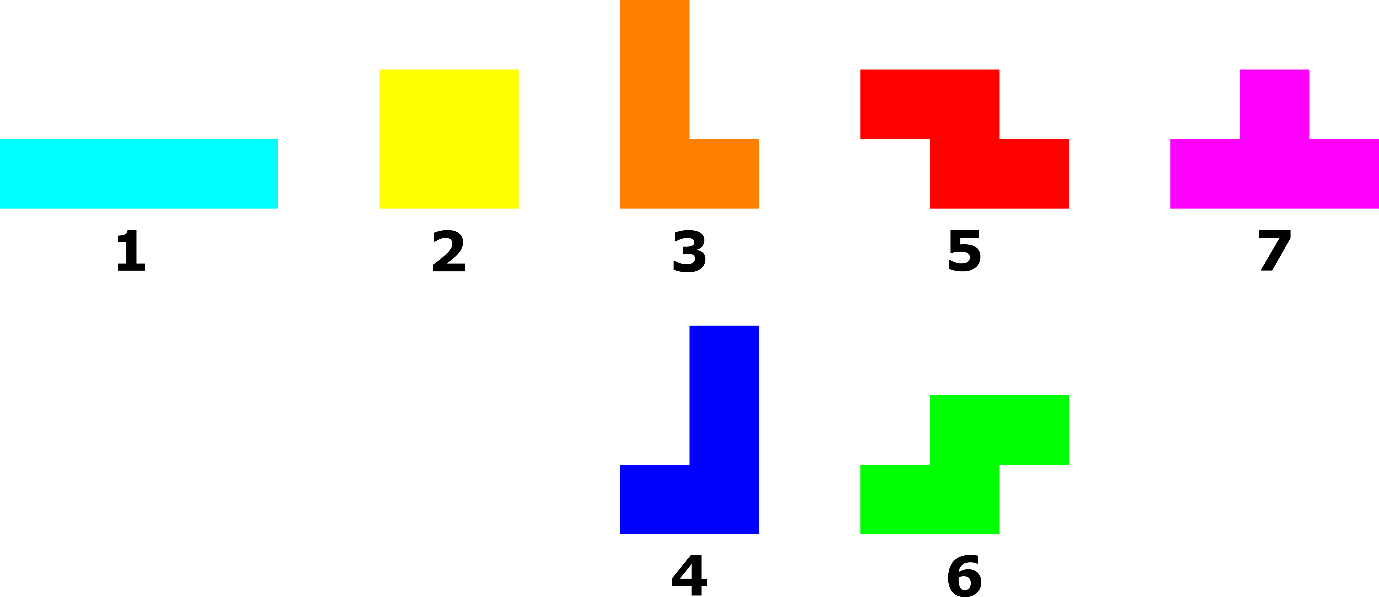


图3：当 时K-联牌的所有类型。四联牌共有7种无法通过平移和旋转重合的类型，分别命名为线性（**1**，linear）、方形（**2**，cubic）、顺式L形（**3**，*cis*-L shaped）、反式L形（**4**，*trans*-L shaped）、顺式Z形（**5**，*cis*-zigzag）、反式Z形（**6**，*trans*-zigzag）和T形（**7**，tee）。

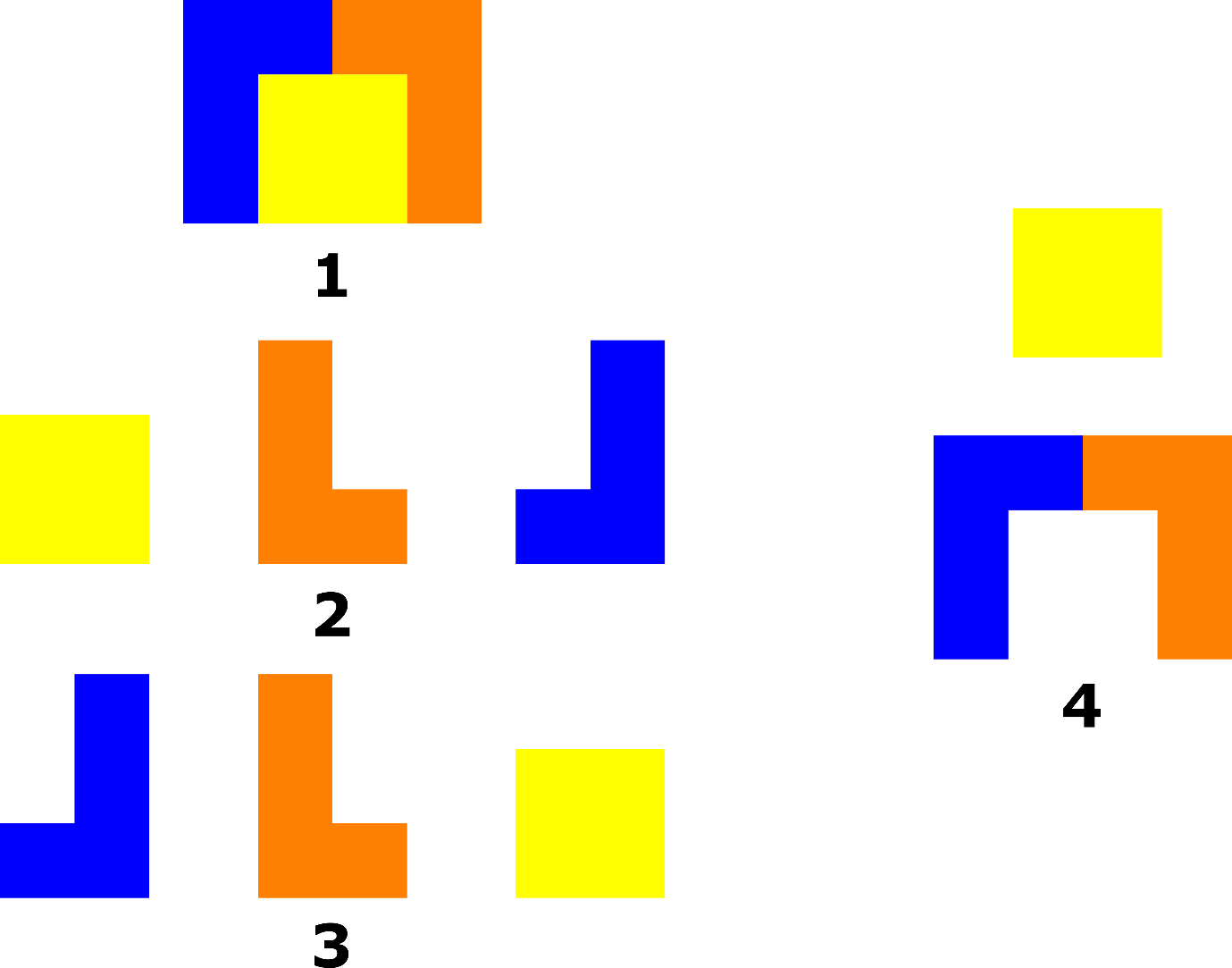


图4：下落顺序的重要性。在 时，若生成算法的第一部分生成了图案**1**，将有 种不同的下落顺序。由于俄罗斯方块游戏规则中阻挡规则的存在，仅有部分合法的下落顺序能够完成对**1**的拼贴，如顺序**2**。其他不合法的顺序会导致被规则禁阻的图案出现，如顺序**3**会导致图案**4**的出现，导致图中的方形四联牌被其他两个四联牌阻挡而无法到达正确位置。

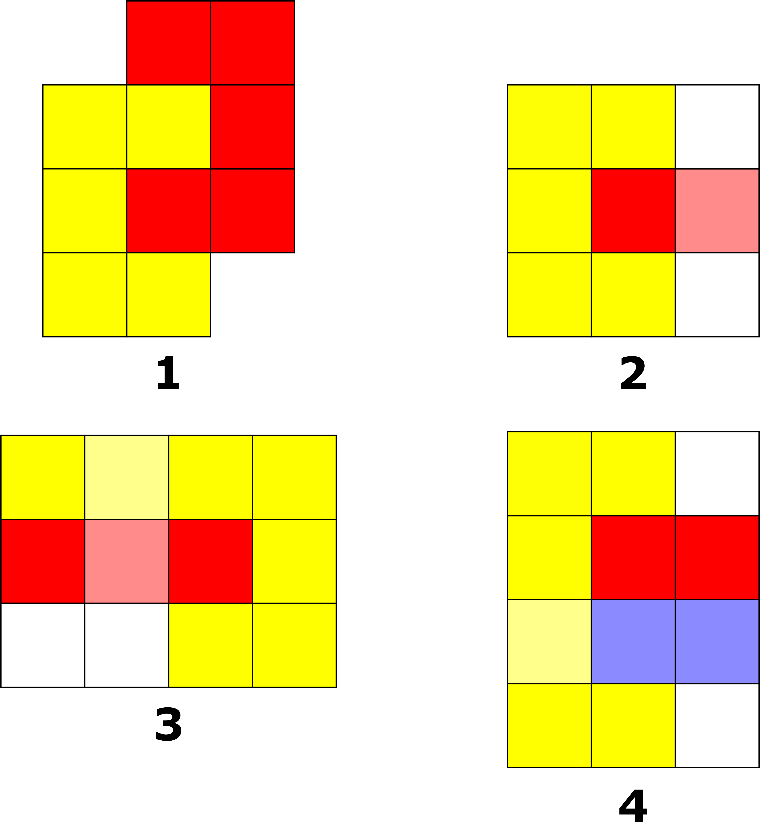


图5：下落顺序的存在性。图中的浅色方格代表一或多个横向的实际方格。在一个 模板**1**中存在循环依赖，红色五联牌依赖于黄色五联牌，而黄色五联牌又反过来依赖红色五联牌。将该二元多联牌循环依赖的模型抽象可以得到**2**和**3**所示的结构。可以看到，当且仅当 时存在该二元循环依赖，因为要形成如**2**所示的“半包围”结构要求一个多联牌至少覆盖5个方格。多元循环依赖可以用**4**描绘。显然，模式**3**和**4**对单个多联牌最小覆盖方格数都比最基础的**2** 大。

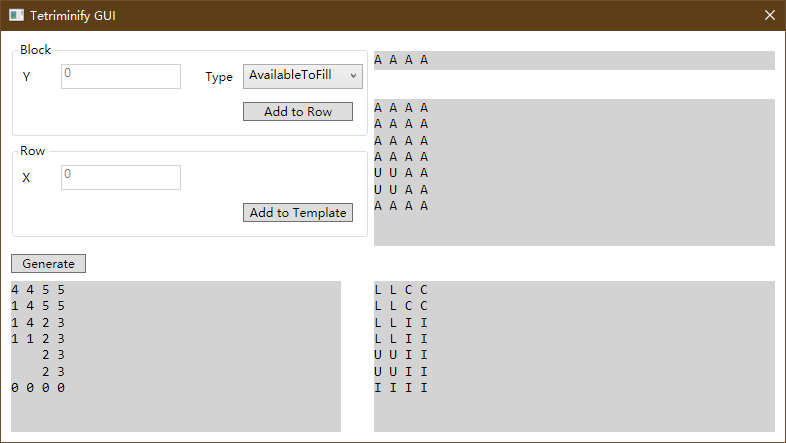


图6：Tetriminify的运行界面。左上方是用户交互区域，用于设置模板空间的内容。右上方是模板空间显示区，“A”代表“available to fill”，“U”代表“unavailable to fill”，分别代表文中提到的1值和0值。右下方显示了四联牌切分算法的输出，字母代表各个四联牌的形状，“U”代表空缺的方格。左下方显示了下落顺序生成算法的输出，标号为0的四联牌将最先下落，空缺部分与模板空间中不可覆盖部分相对应。

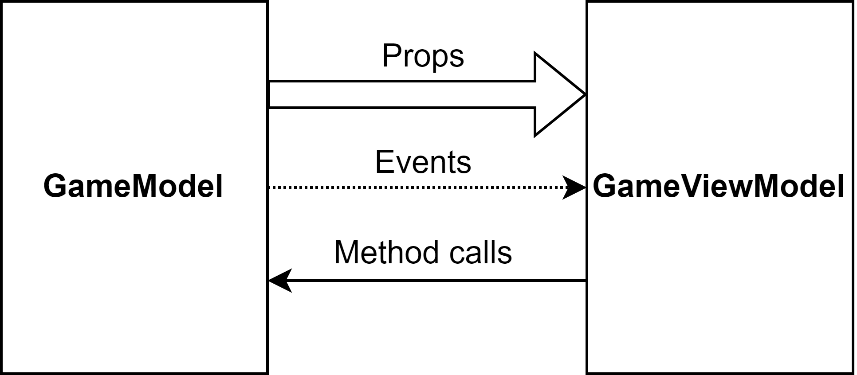


图7：模型与外界的数据交换。模型是对外界高度封装的，有三种单向的数据传递方式：通过读取属性、通过监听事件和通过调用方法。这三种暴露的接口都对信息进行严格的检查，保证模型的数据始终处于可预测的状态，不被意外破坏。

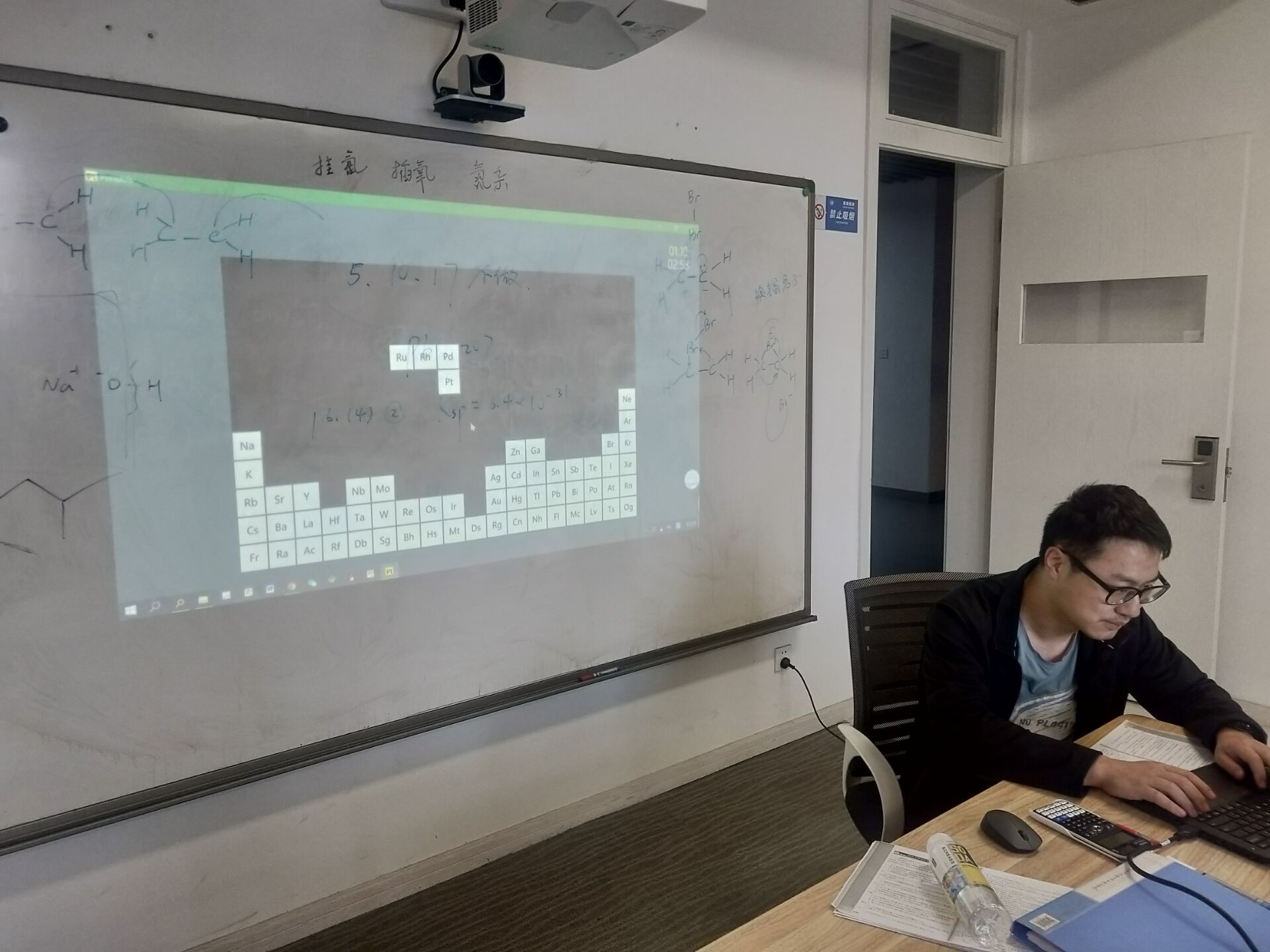


图8：某试用者正在体验Periotris.js的一个早期版本。图片由宁波效实中学徐珑迪老师惠赠。