Experiential Learning in Complex Environments: Simulations using Blockly

Jonathan L. Graves (UBC)[1] & Rik Blok (UBC)[2]

[1]Economics (Faculty of Arts) [2]Integrated Sciences (Faculty of Science)

Contact: jonathan.graves@ubc.ca

INTRODUCTION

Simulations and games are valuable pedagogical tools for engaging students in experiential learning. However, how can we conduct these kinds of experiences in complex environments without compromising either the setting or the learning goals?

- We present a set of tools (software + teaching) which can be used to build and play complex simulations in an accessible and engaging way.

EDUCATIONAL CONTEXT

- *Students*: 2nd year, non-technical, general background (no math or CPSC background).
- *Classroom*: large (100+) classroom, 1 hour sessions, minimal TA support
- *Learning objective*: understand how beliefs can create emergent complexity and cooperative behaviour
- *Experience*: Axelrod (1980)'s repeated Prisoner's Dilemma tournament (RPD)

KEY CHALLENGE

- Experiential learning theory (Kolb, 1980) requires *active experimentation*
  - ➢ Requires on-going interaction with learning context
- Learning context requires (a) complex setting (can't simplify) and (b) experimentation (but no specific skills assumed)
  - ➢ Simulation can be "too complex": learning to "play" the game instead of understanding context
- **How do we let students actively experiment in the RPD without programming skills or tools?**

TEACHING PRACTICE

- Innovation: Educational software based on Blockly framework allows for user-friendly design and playing of simulations
  - ➢ No technical skills necessary
  - ➢ Intuitive, easy to use

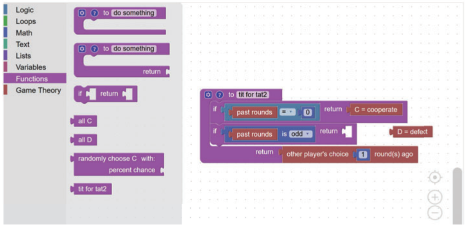
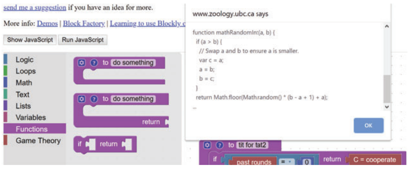


Fig 1: "Strategy Builder" tool

- Generates code which plays simulation (hidden from user)


send me a suggestion if you have an idea for more.

More info: Demos | Block Factory | Learning to use Blockly c

www.zoology.ubc.ca says

```
function mathRandomInt(a, b) {
  if (a > b) {
    // Swap a and b to ensure a is smaller.
    var c = a;
    a = b;
    b = c;
  }
  return Math.floor(Math.random() * (b - a + 1) + a);
...
```

Fig 2: Code generation (JS)

- Student (in groups or solo) build strategies to play RPD, test versus other strategies
  - ➢ Groups recommended for diversity of beliefs in RPD context
  - ➢ Cross-functional (e.g. science + arts) especially recommended
- Instructor conducts and adjudicates final simulation
  - ➢ During design, acts as support and trains students (easy)
- Post-simulation, class examines results and "debriefs" on conclusions, completing the Kolb cycle of experiential learning
  - ➢ Can be repeated or compared with literature/past results

INTEGRATION INTO COURSE

No Requirements Necessary:

- Can use pre-packaged simulations "out of the box" if meets learning context
  - ➢ Currently includes RPD: applications in economics, sociology, biology, mathematics, computer science
- Integrate into classroom setting: get in touch with authors to pilot!

Elementary Programming:

- Program own game theory simulation based on course material
  - ➢ e.g. dictator games, ultimatum games, evolutionary games, etc.
- Mainly need to adjust game settings and Blockly integration (1-2 days)

Intermediate Programming:

- Design your own simulation very flexibly
- Total control over tools and appearance, context
  - ➢ e.g. real-time, procedural, graphics, UI
- Customize all interaction and environments

KEY BENEFITS

- Increased classroom engagement
- Improved mastery of learning objectives

In-context:

- Improved appreciation for diverse beliefs/opinions/experience

REFERENCES

- Kolb, D. A. (2014). *Experiential learning: Experience as the source of learning and development*. FT press.
- Herz, B., & Merz, W. (1998). Experiential learning and the effectiveness of economic simulation games. *Simulation & Gaming*, *29*(2), 238-250.
- Holt, C. A. (1999). Teaching economics with classroom experiments: A symposium. *Southern Economic Journal*, 603-610. Dorn, D. S. (1989).
- Dorn, D. S. (1989). Simulation games: One more tool on the pedagogical shelf. *Teaching Sociology*, 1-18.

Introduction to Blockly (as a Teaching Tool)

What is Blockly?

- Blockly is a free open-source Javascript library originally developed by Google which allows you to build "visual code" for different projects.
- It uses Lego-like "blocks" and an interactive environment which users can interact with to "build" programs (i.e. sets of instructions) to interact with an environment
    - This can be very general (e.g. computer programming) or very specific (e.g. playing a specific game)
- Unlike other similar tools (e.g. Scratch), Blockly allows for complete customization of both the Blocks and the context.
    - This allows instructors to abstract away from "programming" and focus on interaction with the environment
    - It comes with some pre-made blocks for basic programming and math structures (e.g. if-else statements) and a set of tools to help you create your own custom blocks.

How does it work?

- A simulation or environment is a set of rules which define who can act and how those interactions take place. This is usually made into a computer program using a programming language.
- Blockly "lives" in your web-page (or application) and allows users to combine blocks together in pre-defined ways, based on the type of the block (defined by designer).
- The combination of blocks creates output which can interact with the environment through the programming language.
    - All of this is invisible to the end user; they just see the Blocks and what they do!
- A designer (e.g. instructor) can customize the context by:
    - First, creating their own simulation environment which they want to interact with (e.g. writing a program which implements the rules of rock-paper-scissors)
    - Second, customizing a set of blocks which allow users to interact with the simulation (e.g. rock block, paper block, scissor block, a "flip coin")
        - These custom blocks are "told" how the simulation should interpret them
    - Third, setting up a web-page with Blockly which uses the custom blocks to play the game (e.g. Rock Paper Scissors web-page, which players choose blocks to represent their decisions in the game)

What do I need to consider?

- Do I have the skills to set us and use Blockly efficiently?
    - Basic Javascript AOK; some web-design useful.
- Is my simulation complex or interactive enough to need Blockly?
- Do existing tools fit my use-case well enough already?