Identifying Observable Outcomes in Game-Based Assessments

Russell G. Almond Valerie J. Shute Seyfullah Tingir
Seyedahmad Rahimi

Educational Psychology and Learning Systems
College of Education
Florida State University

Maryland Assessment Research Conference, 2018

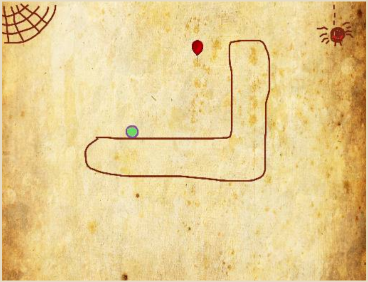
Thanks

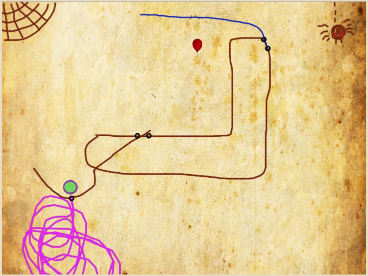
- Bill & Melinda Gates Foundation U.S. Programs Grant Number #0PP1035331
- National Science Foundation, DIP 037988
- Institute of Educational Science, Goal 1 039019

- Fengfeng Ke
- Adam LaMee
- Don Franceshetti*
- Weinan Zhao*
- Xinhao Xu*
- Mathew Martens
- Chen Sun
- Xi Lu
- Ginny Smith
- Curt Fulwider*
- Lukas Liu
- Megan Bayles*
- Buckley Guo*
- Jiawei Li
- Jordan Charles
- Renata Kuba

*Previously involved with the project

Physics Playground





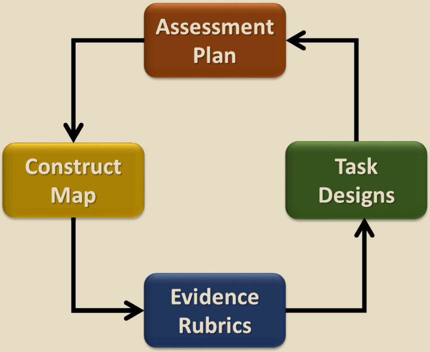
Example Level and Solution from *Physics Playground* Version 1. *Player Goal: Get the ball to the balloon by drawing ramps, levers, springboards and pendulums.*

Project Goals:

- Inference about conceptual physics
- Adaptive Sequencing of Levels
- Adaptive Provision of Learning Supports

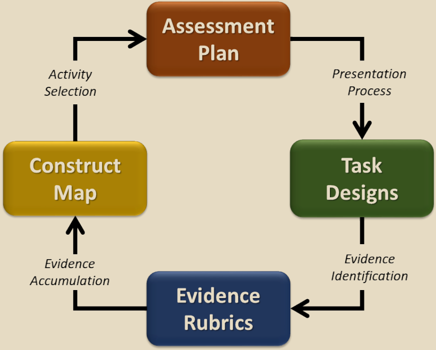
Need to infer state of physics understanding from game logs.

Four Elements of Assessment Design



1. Define a *Construct map* for the skills to be assessed (Proficiency/Competency Model)
2. Describe *evidence* that a student has the constructs (Evidence Model)
3. Create designs for *tasks* where the evidence can be observed (Task Model)
4. Create a (Assessment/Lesson) *plan* for those activities. (Assembly Model)

Four Processes of Assessment Delivery



1. **Presentation Process** — Present the task and log events.
2. **Evidence Identification (EIP)** — Extract key features (*observables*) from the stream of logged events.
3. **Accumulate Evidence (EAP)** — Enter observed outcomes into the measurement model and locate player the construct map.
4. **Activity Selection** — Based on player's current location, select next activity.

Proc4 Messages

```
{
  app: "ecd://epls.coe.fsu.edu/PP",
  uid: "Student 1",
  context: "SpiderWeb",
  sender: "Evidence Identification",
  message: "Task Observables",
  timestamp: "2018-10-22 18:30:43 EDT",
  data:{
    trophy: "gold",
    solved: true,
    objects: 10,
    agents: ["ramp","ramp","springboard"],
    solutionTime: {time:62.25, units:"secs"}
  }
}
```

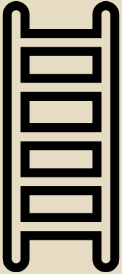
- Generic model of messages passed between processes.
- Application (app) header defines vocabulary used in other fields.
- Context equals task in this example.
- Data field can hold any number/kind of object.

Four Processes in Physics Playground

- *Presentation Process* — Client–server system which displays game levels and learning supports.
    - Written in Unity, with videos linked from YouTube.
    - Logs events in Learning Locker® learning record store.
    - Identifies agents of motion from drawings (ramp, springboard, lever, pendulum)
- *Evidence Identification* — Needs to generate a list of *observables* from the learning record store.
    - *New system needed for this.*
- *Evidence Accumulation* — Bayesian Network takes observables and updates probabilities that student has skills
    - Written in Peanut and Netica® `https://pluto.coe.fsu.edu/RNetica`.
- *Activity Selection* — Uses expected weight of evidence to indicate next level to use.

ECD design objects used for *Physics Playground*

Construct Ladders



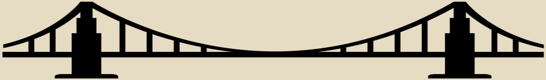
- Construct is implicitly defined through tasks.
- Construct Map (Wilson, 2006). For High, Med and Low levels:
    - What kind of *tasks*?
    - What qualities of *performance*?
    - What *personal* qualities?
- Evidence column: provides ideas for how to test.
- Complete definition includes multiple ladders and relationship between them.
    - Competencies are usually defined hierarchically (Math and Science).
- Target population is important.
- *Joint effort of Experts and Design Team*

Variable Definition Spreadsheet

Table 1: ES Spreadsheet (Excerpt)

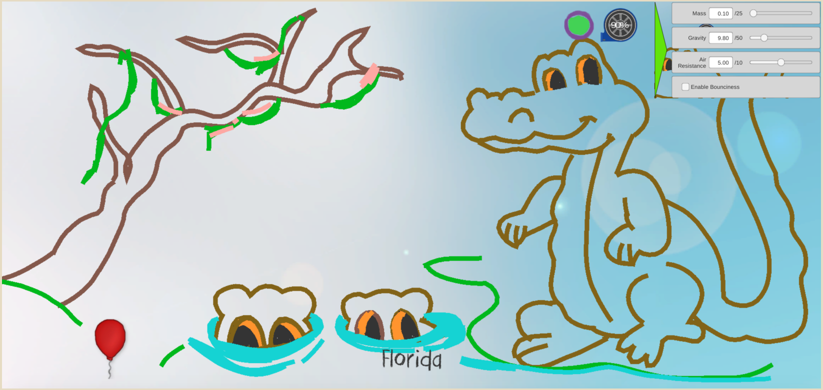
Competency	Sub-competency	Description	Evidence
Force and Motion	Newton's 1st Law	Static equilibrium (a=0 and v=0)	Player applies or adjusts a force (e.g., nudge, blow, gravity, air resistance) to keep an object stationary in at least one dimension.
Force and Motion	Newton's 2nd Law	Net force and acceleration are directly related	Player applies or adjusts a force acting on an object to cause it to accelerate at a desired rate.
Linear Momentum	Properties of momentum	Momentum is directly related to mass	Player adjusts the mass an object to affect the amount of momentum it transfers to a second object after the two collide.
Energy	Energy can transfer	Energy can transform from one type to another (e.g., GPE to KE)	Player changes parameters (e.g., mass, position, speed) to transform more or less energy of one type to another (e.g., KE, GPE, EPE) of the same object.
Torque	Properties of torque	Force and torque are directly related.	Player adjusts the magnitude of a force to cause a corresponding change in the magnitude of a torque.
Science and Engineering Practices	Use iterative design to solve a problem	Solve a problem by making variations on previous strategies	Player makes successive adjustments of the same parameter to solve a level.

Evidence: Data linked to a Ladder



- Define *observables* which are links between EIP and EAP.
- Link observables to constructs in competency model, using Q-matrix.
- Specify rules for how observable values can be set from event log data.
- Check for coverage of the constructs in the assessment.
  - Discovered we didn't have enough coverage in Version 1!

Manipulation Levels

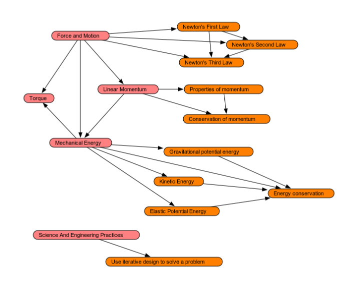


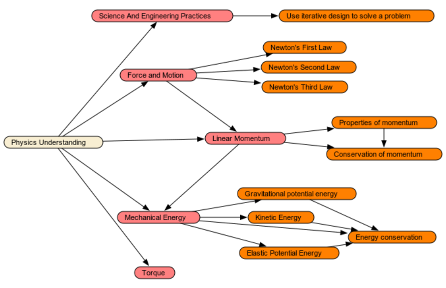
Goal: Get the ball to the balloon by manipulating the ball's mass, gravity, air resistance, and/or bounciness.

Learning Supports

- Version 2 includes explicit learning supports
- Show me the Physics
  - Applicable across multiple levels
  - Interactive Physics Definitions
  - Hewitt Videos
  - Physics Animations
- Show me the Solution
  - Level Specific Help
  - Hints: e.g., “Try drawing a lever.”
  - Worked Examples
- Game control reviews

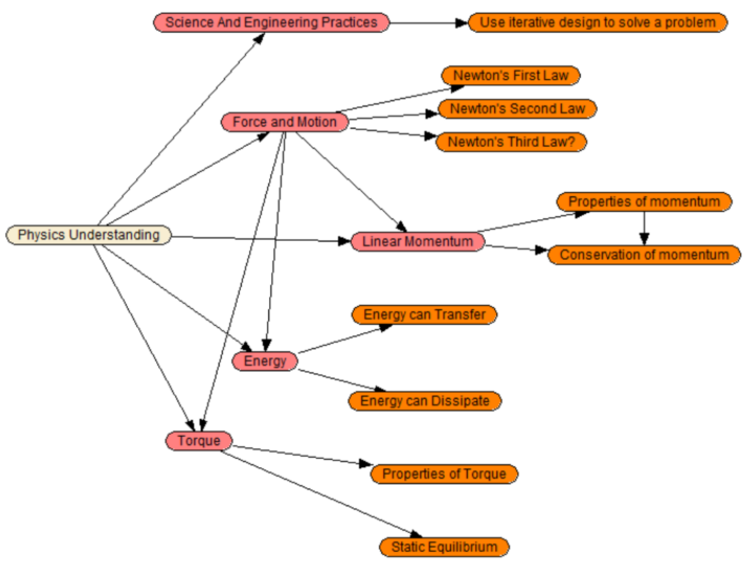
Two Candidate Construct Models





- Gave experts two choices of construct model
- Discused alternatives

Final (Version 6) Structure



Expert-Based Conditional Probability tables

- Used DiBello (IRT-like) parameterizations.
- Modeling Team produced first pass numbers based on some general guesses.
- Want Physics experts to review work, but they are not familiar with our notation.
- Want to deliberately anchor experts towards high correlations between latent variables.
- Solution: Translate into natural language for expert review.
- Also, show table along with parameters so expert can see effect.
- Should have again given experts choices to select among.

Defining Observables

- List possible observables associated with each level.
- Link observables to levels in a spreadsheet.
- Produce operational definitions for observables:
  - When is a drawn object considered a lever? (This requires physics engine, so done in presentation process)
  - What constitutes an "attempt" if the observable is number of attempts? (This is done in evidence identification process)
- Set cut scores for counts and continuous variables used as ordinal variables in Bayes nets.
  - What are long, medium, and short completion times for level?
  - What are low, medium, and high number of manipulations for a slider?
  - These values will be linked to levels in Bayes net (so implemented in the Evidence Accumulation process)

Sample Observables

- Obs 1 What is the maximum value coin (gold, silver or none) that the player has earned for this level?
- Obs 2 Did the player manipulate the gravity slider?
- Obs 3 How many objects did the player draw?
- Obs 4 How much time (excluding time spent on learning supports) did the player spend on the level?
- Obs 5 Did the player attempt to draw a springboard?

- Obs 1 and 2 can be read directly from event stream.
- Obs 3 and 4 require keeping track of state of system.
- Obs 5 requires detailed knowledge of Physics engine: put in presentation process.

Learning Record Store

```
{
  app:"ecd://coe.fsu.edu/EPLS/
      AssessmentName",
  uid:"Student/User ID",
  timestamp:"Time of event",
  verb:"Action Keyword",
  context: "Context ID",
  object:"Object Keyword",
  data:{
   field1:"Value",
   field2:["list", "of", "
       values"],
   field3:{part1:"complex",
       part2:"object"}
  }
}
```

- Simplified version of Learning Locker's xAPI format.
- Application (app) controls vocabulary for other fields.
- Verb and Object fields define what the event is.
- Context might be supplied in presentation or evidence identification.
- Data do not have defined schema.
- Stored in a document database (Mongo)

Determining Tasks from Event Traces

- *PP* engine does not include game level in context.
    - Events for level started/restarted.
    - Events for level successfully completed.
    - Events for level exited.
- If *task=game level*, then task must be inferred from event log.
- In four process model *task* was the granularity at which information was sent around cycle.
- In a simulator, a task may have many scoreable units (*contexts*).

Tasks versus Contexts

- Consider a Flight Simulator
- Task is fly from one airport to another.
- Embedded Contexts:
  1. Pre-flight Checks
  2. Take-off
  3. Cruising
  4. Thunderstorm
  5. Cruising
  6. Approach
  7. Landing
- Context is inferred from state of system.
- Evidence Identification tracks state of system to determin context

Tracking System State

```
{
  uid: "Identifier",
  context: "Task Name",
  oldContext: "Task Name",
  timers:{...},
  flags:{...},
  observables:{...}
},
```

- Timers are used to track time spent in various states
  - `state.timers.`*name*
- Flags are used for variables that won't be reported.
  - `state.flags.`*name*
- Observables are reported
  - `state.observables.`*name*
- Old Context allows noting when context has changed.
  - `state.context`, `state.oldContext`,

Types of Observables

- Final Observables (observables)
  - Sent to evidence accumulation (EAP)
  - Used for context-level feedback
  - Logged for research purposes
- Intermediate Observables (Timers and Flags)
  - Used to calculate final observables
- ETS's e-Rater®
  - Intermediate observables count various writing errors
  - These are aggregated into Grammar, Usage, Mechanics and Style final observables
  - Final observables are added to regression model to get final score

Rules of Evidence

```
{
  ruleName:"Human readable
      identifier",
  doc:"Human language
      description",
  context:"Context or Group
      Keyword",
  verb:"Action Keyword or
      ALL",
  object:"Object Keyword or
      ALL",
  ruleType:"Type Keyword",
  priority:"Numeric Value",
    condition:{...},
    predicate:{...}
}
```

- Rules must match on verb, object and context to be applicable.
  - Context groups allow rules to be used in many contexts (manipulation tasks)
- Conditions determine when rules fire
- Predicates define rule action
- Type and Priority are used for sequencing.

Rule Types and Priority

- Five Types of Rules are executed in sequence:
  1. *State Rules*—Update flags and timers.
  2. *Observable Rules*—Update observables.
  3. *Context Rules*—Check for changes in contexts
  4. *Trigger Rules*—Send messages to other processes.
  5. *Reset Rules*—Clean up state when context changes.
- Rule priority determines sequence within a type.
  - Lower numbers first.
  - Resolves conflicts
  - Satisfy preconditions

Conditions

```
condition:{
  <field1>:<value1>,
  <field2>:[<value-list>],
  <field3>:{"?op":<value3>},
  "?where":<function>,
  ...
}
```

- Basic test is if field has specified value, or in list of values.
- Field can refer to state or event (as can value).
- Op list includes:
  - `?eq`, `?ne`, `?gt`, `?gte`, `?lt`, and `?lte`
  - `?in`, `?nin`
  - `?exists`, `?isnull`, `?isna`
  - `?any`, `?all`
  - `?not`, `?and`, `?or`
  - `?regex`
- `?where` allows calling arbitrary R functions

Predicates

```
predicate:{
   <update operator1>:{ <field1>: <value1>, ... },
   "!set":{ state.flags.<logical>: true},
   "!set":{ state.timers.<name>.running: true},
   "!incr":{ state.flags.<count>: 1},
   "!setCall":{ state.flags.<name>: <function>},
   ...
}
```

- Predicates only change fields in state
  - Can reference data in event
- A number of different modification operators.
  - `!set`, `!unset`
  - `!incr`, `!decr`, `!mult`, `!div`, `!min`, `!max`
  - `!addToSet`, `!pullFromSet`, `!push`, `!pop`
- For timers can set `state.timers.`*`name`*`.running` `state.timers.`*`name`*`.time`
- `!setCall` runs R code.

Extended Example: Counting Air Slider Manipulations

Initial State:

```
{
  uid: "Test0",
  context: "Air Level 1",
  timers:{},
  flags:{
    airUsed: true,
    airOldVal: 17,
  },
  observables:{
    airManip:1
  }
}
```

Event:

```
{
  app:"https://epls.coe.fsu.
      edu/PPTest",
  uid: "Test0",
  verb: "Manipulate",
  object: "Slider",
  context: "Air Level 1",
  timestamp:"2018-09-25 12:1
     2:28 EDT",
  data: {
    gemeObjectType: "
        AirResistanceValueManipu
        ",
    oldValue: 0,
    newValue: 5,
    method: "input"
  }
}
```

Extended Example (pt II)

Rule:

```
{
  name: "Count Air Resistance Manipulations",
  doc: "Increment counter if slider changed.",
  verb: "Manipulate",
  object: "Slider",
  context: "Manip Lvls",
  ruleType: "Observable",
  priority: 5,
  conditions: {
    event.data.gemeObjectType:"AirResistanceValueManipulator
        ",
    event.data.oldValue:{"?ne":event.data.newValue}
  },
  predicate: {
    "!incr":{state.observables.airManip:1}

  }
}
```

Extended Example (pt III)

Result:

```
{
  uid: "Test0",
  context: "Air Level 1",
  timers:{},
  flags:{
    airUsed: false,
    airOldVal: "NA"
  },
  observables:{
    airManip:1
  }
}
```

Reflections

- ECD, particularly the Q-matrix, is useful for managing development work.
- It is better to give experts two choices and have them pick.
- Building evidence rules with their test sets.
- Evidence Rules can be sorted
  - By Observable
  - By Event

Software Frameworks

- Game demo and level editor: `https://pluto.coe.fsu.edu/ppteam/pp-links`
- Peanut and RNetica `https://pluto.coe.fsu.edu/RNetica`
- Proc4, EIEvent and EABN `https://pluto.coe.fsu.edu/Proc4`
  - Still in early development.