project

- **goal**: get computer science into middle schools
- **approach**: scalable game design to motivate and educate students to learn about computer science through game design
- **research**:
  - equity: gender, underrepresented (Native Americans)
  - transfer to STEM: “now that you can make the game, can you make a science simulation?”
  - computational thinking (an NSF priority)
- **scope**:
  - years 1-3: strategic exploration of Colorado spectrum
  - years 4-8: scale up to US.
“programming, oh no... I know what is going to happen. The teacher writes a program onto the blackboard, we type it into the computer and it never works” — student
our goal: bring computer science into public schools

- most ITEST programs are after school programs
- motivate and education through scalable game design
- support computational thinking
- facilitate computational science
- provide access to ALL students including women and minorities, e.g., BVSD “forced elective”
4 areas

- Tech Hub: Boulder, BVSD
- Inner City: Aurora
- Rural: Pueblo
- Remote/Tribal: Ignacio, CO, and Oglala, SD
a LARGE project

Oglala, SD to Ignacio, CO
715 mi – about 12 hours 40 mins drive time

1120 students

Boulder (co grad rate: 36%)

Oglala (co grad rate: 1.6%)

Pueblo (co grad rate: 11%)

Aurora (co grad rate: 18%)

Ignacio (co grad rate: 11%)
process: middle school

middle school: 6th Grade

motivational focus: computer science is interesting; I can do this

required
game design
module
1 week

all 6th grade students

trained middle school teacher

trained community/tribal community student

teach k-12 students how to make a simple frogger-like game, administer motivational questionnaires and organize family participation

family show & tell

motivational survey

middle school: 7th or 8th Grade

STEM focus: game design connections to math & science, computational science

elective
game design
module
4 weeks

self selected 7th grade students

trained middle school teacher

trained community/tribal community student

teach k-12 students how to make sophisticated games and simulations including math and science, administer motivational questionnaires and organize family participation

family show & tell

motivational survey
Process: High School

High School

Self selected high school students

Motivational survey

Elective transition
1 week

Elective
Game design using Java, C#, Flash
8 weeks

Trained high school teacher

Teaches K-12 students how to use programming languages such as Java, and C#; helps students making transition from visual tools

Computer science skills
Focus: Software engineering, programming, object oriented design

The Real World

University

On the job training

Community/tribal college

Workforce
history
AgentSheets

- Be able to express and communicate complex ideas
- Education
  - History
  - Math
  - Music
  - Geology
  - Logic
  - Programming
  - Art
AgentSheets92: rewrite rules

- creating rules for agents via programming by example
- descendent systems: Apple KidSim, Cocoa, Stagecast Creator
- Problem: trapped by affordances
AgentSheets95: Tactile Programming

- use drag and drop interfaces to
  - compose: avoid syntax problems
  - debug: incremental testing
  - share: e.g., Web-based Behavior Exchange
- descendants: Squeak, eToys, Alice, Scratch
future
AgentCubes

- A radically new kind of computational thinking tool for 2D-3D game and computational science applications

- **approach**: an incremental 3D approach to create tool to create 3D
  - models
  - animation
  - programs
  - visualization
2D or 3D? YES
inflatable icons
computational thinking tool checklist

for systemic impact a computational thinking tool used in K-12 must fulfill *all* these conditions:

1. *has low threshold*: can make a game in first 2 hours
2. *has high ceiling*: can make a real game that is playable and exhibits sophisticated behavior, e.g., complex AI
3. *scaffolds Flow*: provide educational game design stepping stones with managed skills & challenges
4. *is multipurpose*: works for game design and computational science applications
5. *supports equity*: accessible and motivational across gender and ethnicity boundaries
6. *systemic & sustainable*: can be used by *all* teachers to teach *all* students. Support: teacher training, standards alignment
#1 has low threshold
simple problem ➔ simple solution?

Frogger:
user can move frog (up, down, left, right) with cursor keys
benchmark: frogger in 3 hours

authoring at cognitive level:
most or all components in the
solution domain map back to
problem domain

- simple ideas ⇒ simple
  formalization
- intrinsic complexity

authoring at traditional
programming level: many
components in solution
domain have no meaning in
problem domain.

- ideas ⇒ complex code
- accidental complexity
Many tools claim to enable students to build games. If tools fall short in building even simple games that their creators and their friends like to play then excitement quickly fades.

A game is more than colorful objects moving around on the screen making noises and speech bubbles.

If the tool cannot handle (hard ... impossible) fundamental computational thinking patterns such as transportation (e.g., Alice) then you cannot build even simple 1980 arcade style game such as Frogger.
#2 has high ceiling
“excuse me, I need better AI”

- excited students want to move on. The tools need to be able to create advanced games with sophisticated AI
  - can you find the shortest path in a maze?
  - can you have collaborative agents?
  - can you model human motivation?
STEM in games

game world

sophisticated visualizations

$$u_{0,t+1} = u_{0,t} + D \sum_{i=1}^{n} (u_{i,t} - u_{0,t})$$

advanced math (diffusion)
transition to traditional programming

```java
public class frog extends ActiveAgent {
    public synchronized void tasks() {
        if (key(126)) {
            move((byte)1);
            return;
        }
        if (key(123)) {
            move((byte)3);
            return;
        }
        if (key(124)) {
            move((byte)5);
            return;
        }
        if (key(125)) {
            move((byte)7);
            return;
        } else {
            return;
        }
    }
```
#3 scaffolds

Flow
FITness flow
gentle slope
curriculum
basic computational thinking patterns

- **Collision**: Frogger: Frog meets Truck
- **Push**: Sokoban: person pushes boxes
- **Transport**: Frogger: logs transport frogs
- **Generate**: Space Invaders: defenders shoot rockets
- **Absorb**: Bridge Builder: tunnel absorbs cars
- **Choreography**: Space Invaders: mother ship makes attack alien ships move left and right and descend
- **Polling / Counting**: Pacman: game over when all the dots are eaten
advanced patterns

- **Diffusion**: Electricity, Heat, rumors, toys: spread of information
- **Seeking**: Sims: people finding food
- **Collaborative Diffusion**: Soccer: players collaborate and compete
- **Multiple Needs**: Maslow’s hierarchy of needs
#4 scalable game design

is

multipurpose
fundamental question of transfer

- school district leaders – barely tolerate game design but want to see transfer of computational thinking skills to STEM (Science Technology Engineering Math)
- “now that you can make Space Invaders can you make a scientific simulation of a mud slide?”
- bold claim: if there is not transfer, there is no computational thinking
computational science

- if your math operators are limited to elementary school math you will not get far in STEM
- if you have no plotting and visualization tools you have no way to analyze computational science models

\[ \sin \text{(time)} \times 0.1 \times \text{value[left]} + 0.9 \times \text{value} \]
computational science model
computational thinking inventory
#5 supports equity
Formal studies, such as an independent research study by the Stanford School of Education, illustrate that novices report high level of desire to continue with AgentSheets. In fact, both boys and girls expressed the same high levels desire to continue using AgentSheets.

“I used to only have 2 or 3 girls in my elective classes, now half of the class is girls.”

– Greg Peters, Applied Tech, Computer, Monarch K-8, with participation of 80-100 students per trimester.
“A year ago, the boy could barely read. And now he's doing OOP [object oriented programming]--I love that”

– David Brode, father of middle school student using AgentSheets
first US-wide Native American Game Design class

- summer of 2008
- course filled up immediately
- more women than men
#6 is systemic & sustainable
elective -> required

- computer science education should not be like a field trip to the zoo!
- research needs to move beyond self selected teachers teaching self selected students:
  - crossing the chasm: grassroots individual school -> entire school district
  - ALL teachers
  - ALL students
- examples
  - entire school district: Boulder Valley School District: 12 middle schools
  - entire country: Greece introducing now
need to integrate with standards

ISTE NETS <-> Scalable Game Design

Creativity and Innovation
Students demonstrate creative thinking, construct knowledge, and develop innovative products and processes using technology. Students:

- **apply existing knowledge to generate new ideas, products, or processes:**
  - design and develop games
  - design and develop computational science models

- more: [http://scalablegamedesign.cs.colorado.edu/wiki/Standards](http://scalablegamedesign.cs.colorado.edu/wiki/Standards)
computational thinking tool checklist

We are at the tipping point of making systemic impact. For systemic impact a computational thinking tool used in K-12 must fulfill all these conditions:

1. has low threshold
2. has high ceiling
3. scaffolds Flow
4. is multipurpose
5. supports equity
6. systemic & sustainable
Computational Thinking

“Drama is life without the boring bits”
— Alfred Hitchcock

“Computational Thinking is programming without the boring bits”

computational thinking

programming

nested loops

magic (pixel) constants