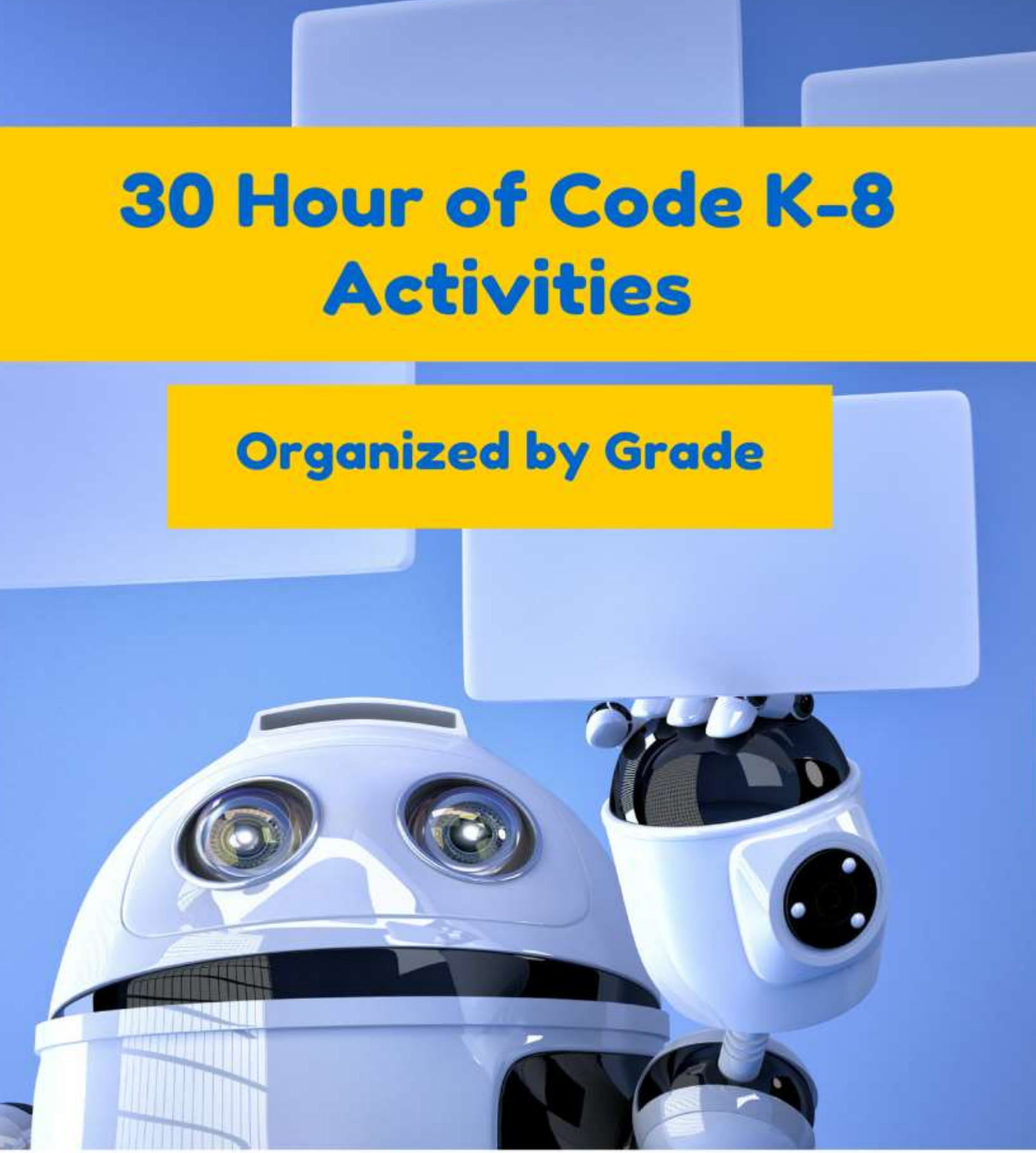
30 Hour of Code K-8 Activities

Organized by Grade



by Ask a Tech Teacher

30 Hour of Code K-8 Activities

Organized by Grade

Kindergarten
- *Online activities*
- *Misc. coding websites*
- *Human robot*
- *Human algorithm*

1st Grade
- *Online activities*
- *Misc. coding websites*
- *Human robot*
- *Human algorithm*

2nd Grade
- *Online activities*
- *Misc. coding websites*
- *Sequencing*
- *Animation*

3rd Grade
- *Online activities*
- *Coding pixel art*
- *Misc. coding websites*

4th Grade
- *Find language-specific symbols*
- *Using Alt codes*
- *Programming shortkeys*
- *Hour of Code lessons*
- *Misc. coding websites*

5th Grade
- *Program macros*
- *Program shortkeys and hotkeys*
- *Online Hour of Code lessons*
- *Visit miscellaneous websites*

6th Grade
- *Build an app*

7th Grade
- *Scratch*
- *Auto Hotkeys*
- *Wolfram/Alpha*

8th Grade
- *Alice*

6th-8th Grade
- *SketchUp*

Table of Images

Figure 1a-b--What people think programming is... ..... ***Error! Bookmark not defined.***
Figure 2a-f--Human sequence ..... ***Error! Bookmark not defined.***
Figure 3a-i--Human algorithm ..... ***Error! Bookmark not defined.***
Figure 4a-b—What coding looks like ..... ***Error! Bookmark not defined.***
Figure 5a-b—Kindergarten coding ..... ***Error! Bookmark not defined.***
Figure 6a-f—Human robot parts ..... ***Error! Bookmark not defined.***
Figure 7a-i—Human algorithm ..... ***Error! Bookmark not defined.***
Figure 8a-b: Which is programming? ..... ***Error! Bookmark not defined.***
Figure 9a—Kodable; 9b—Hopscotch; 9c—Tynker ..... ***Error! Bookmark not defined.***
Figure 10a-c: Correct sequence ..... ***Error! Bookmark not defined.***
Figure 11a-c—Stick figure animation ..... ***Error! Bookmark not defined.***
Figure 12a-b—Coding ..... ***Error! Bookmark not defined.***
Figure 13a-d—Coding in K-2 ..... ***Error! Bookmark not defined.***
Figure 14a-c Pixel art ..... ***Error! Bookmark not defined.***
Figure 15a-c—K/1 Spreadsheet drawings ..... ***Error! Bookmark not defined.***
Figure 16--Thinking hard poster ..... ***Error! Bookmark not defined.***
Figure 17a-b—Programming ..... ***Error! Bookmark not defined.***
Figures 18a-d—Coding in K through 3rd grade ..... ***Error! Bookmark not defined.***
Figure 19—Popular unusual shortkeys ..... ***Error! Bookmark not defined.***
Figure 20—Create a shortkey ..... ***Error! Bookmark not defined.***
Figure 21a-b—What programming feels like vs. what it is ..... ***Error! Bookmark not defined.***
Figure 22a-d—Coding from previous years ..... ***Error! Bookmark not defined.***
Figure 23—How to create a macro ..... ***Error! Bookmark not defined.***
Figure 24—How to create a shortkey ..... ***Error! Bookmark not defined.***
Figure 25a-b—What programming feels like vs. what it is ..... ***Error! Bookmark not defined.***
Figure 26a-e—Coding from previous years ..... ***Error! Bookmark not defined.***
Figure 27a-b--What programming feels like vs. what it is ..... ***Error! Bookmark not defined.***
Figure 28a-e—Coding from previous years ..... ***Error! Bookmark not defined.***
Figure 29—Scratch program page ..... ***Error! Bookmark not defined.***
Figure 30a-b—Scratch script and result ..... ***Error! Bookmark not defined.***
Figure 31—Scratch tools I ..... ***Error! Bookmark not defined.***
Figure 32—Scratch tools II ..... ***Error! Bookmark not defined.***
Figure 33a-b—Scratch remix ..... ***Error! Bookmark not defined.***
Figure 34a-d—Scratch projects ..... ***Error! Bookmark not defined.***
Figure 35—Scratch embed ..... ***Error! Bookmark not defined.***
Figure 36a-b—Blogs about programming ..... ***Error! Bookmark not defined.***
Figure 37—Scratch rubric ..... ***Error! Bookmark not defined.***
Figure 38—Wolfram/Alpha widget ..... ***Error! Bookmark not defined.***
Figure 39a-b—Wolfram/Alpha completed widget; 39c—embedded in blog ..... ***Error! Bookmark not defined.***
Figure 40a-b--Class using Alice ..... ***Error! Bookmark not defined.***
Figure 41a--Student using Alice; 41b—first world ..... ***Error! Bookmark not defined.***
Figure 42a—Make sense of problems; 42b—reason abstractly ..... ***Error! Bookmark not defined.***
Figure 43—Construct viable arguments (in Alice) ..... ***Error! Bookmark not defined.***
Figure 44a—The model; 44b—the result in Alice ..... ***Error! Bookmark not defined.***
Figure 45—Use appropriate tools (Alice) ..... ***Error! Bookmark not defined.***
Figure 46a—Attend to precision (in Alice); 46b—look for and express regularity ..... ***Error! Bookmark not defined.***
Figure 47a-b—Alice programming ..... ***Error! Bookmark not defined.***
Figure 48a-c--Math programming in Alice ..... ***Error! Bookmark not defined.***

Figure 49—Alice rubric ..... ***Error! Bookmark not defined.***
Figure 50a-c—Designs from SketchUp Warehouse ..... ***Error! Bookmark not defined.***
Figure 51a-d—Geometric shapes in SketchUp ..... ***Error! Bookmark not defined.***
Figure 52—House in SketchUp ..... ***Error! Bookmark not defined.***
Figure 53a-b: Which is real? Which is SketchUp? ..... ***Error! Bookmark not defined.***
Figure 54a—Icosahedron in SketchUp; 54b—building on campus ..... ***Error! Bookmark not defined.***
Figure 55a—Ancient Rome; 55b—molecules; 55c—math shapes ..... ***Error! Bookmark not defined.***

Second Grade—Four Activities

Vocabulary	Problem solving	Skills
• *animation* • *coding* • *debug* • *Hour of code* • *Programming* • *screenshot* • *sequence* • *symbolism*	• *I don't know how to use the programming tool (experiment; be a risk-taker)* • *I don't like coding (why?)* • *My partner does lots of the work (that's OK if you do your part)* • *I couldn't debug my program (start at the beginning)*	**New** *Coding/programming* *Animation* *Screenshots* **Scaffolded** *Problem solving*
Academic Applications *Math, critical thinking, habits of the mind*	**Materials Required** *Coding links, membership in onsite program (i.e., Code.org)*	**Standards** *CCSS Stds for Math. Practice* *NETS: 4b, 6c*

Essential Question

How do I use a program I've never seen before?

Big Idea

By thinking critically, I can create something new and useful.

Teacher Preparation

- Have all coding/programming tools ready to use.
- Talk with grade-level team so you tie into conversations.
- Know which tasks weren't completed last week and whether they are necessary to move forward.
- Integrate domain-specific tech vocabulary into lesson.
- Try to get additional time—at least 75 minutes—with each class to complete the activities.

Assessment Strategies

- *Followed directions*
- *Anecdotal observations*
- *Joined class conversations*
- *[tried to] solve own problems*
- *Worked well with a partner*
- *Made decisions that followed class rules*
- *Left room as s/he found it*
- *Higher order thinking: analysis, evaluation, synthesis*
- *Habits of mind observed*

Steps

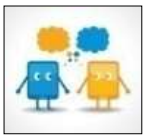
_____Discuss critical thinking and problem solving. Does this apply to, say, Minecraft?

_____The reasons educators embrace coding are simple: **It teaches children to think.** Discuss fundamental programming concepts:

- ***abstraction and symbolism****–variables are common in math, but also in a student's education. Tools, toolbars, images–these all represent something bigger. See this post on the symbolism of the word 'Turkey' (http://wp.me/pZUgb-2wS).*
- ***creativity****–think outside the box; develop solutions no one else has*
- ***debugging****–write-edit-rewrite; when you make a mistake, don't give up or call for an expert. Look at what happened and fix where it went wrong.*
- ***if-then thinking****–actions have consequences*
- ***logic****–go through a problem from A to Z, understand the predictability of movements*
- ***sequencing****–know what happens when; mentioned in CCSS for grades 1 through 5*

_____Share this with students and get their thoughts:

"In 1997, the New York Times reported, 'It may be a hundred years before a computer beats humans at Go.' It took 16 years."



_____December will host the **Hour of Code**, a one hour introduction to coding, programming, and why students should love it. It's designed to demystify code and show that anyone can learn to be a maker, a creator, and an innovator.

_____Coding is a great tie-in to Common Core Math Standards. Any time you show students how to use math skills outside of math, it surprises them. They don't expect a discussion on problem solving or modeling to come from math.

_____Review the Common Core Standards for Mathematical Practice. If you are not a Common Core school, review the similar guidelines from your Standards:

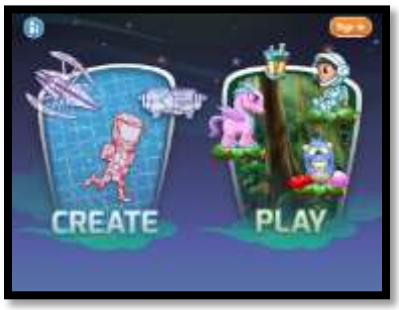
- CCSS.Math.Practice.MP2
  Reason abstractly and quantitatively
- CCSS.Math.Practice.MP3
  Construct viable arguments; critique reasoning of others
- CCSS.Math.Practice.MP4
  Model with mathematics
- CCSS.Math.Practice.MP5
  Use appropriate tools strategically
- CCSS.Math.Practice.MP6
  Attend to precision
- CCSS.Math.Practice.MP7
  Look for and make use of structure
- CCSS.Math.Practice.MP8
  Look for and express regularity in repeated reasoning

_____Most students think programming looks like *Figure 8a* when it actually looks like *Figure 8b*:

Figure 1a-b: Which is programming?

_____Do students remember coding activities from previous years—*Figures 9a-c:*

Figure 2a—Kodable; 9b—Hopscotch; 9c—Tynker



_____This lesson presents four approaches to coding. Pick one that works for your student group:

- *Hour of Code lessons*
- *miscellaneous coding websites*
- *sequencing*
- *animation*

_____These can be done individually or in small groups. They may be done any time during the school year.

Follow one of the free online Hour of Code programs

_____Websites like Code.org (Google for web address) offer full lesson plans for Hour of Code. This is the easiest way to get involved in programming as they do all the planning for you. This may be exactly what you need.

_____Before starting, review digital citizenship–especially privacy.

Miscellaneous coding websites

_____Here are some great coding/programming websites 2nd graders find exciting:

- Build with Chrome
  https://www.buildwithchrome.com/
- Minecraft
  http://minecraftedu.com/
- ScratchJr.
  http://www.scratchjr.org/
- Tinkercad–3D modeling–fee–perfect for 3D printing
  https://www.tinkercad.com/

_____Here's a list of coding websites, by grade level (http://bit.ly/1TbyCIa).

_____Here are apps that take coding to iPads if you're a 1:1 iPad school:

- App Inventor–build Android apps on a smartphones
  http://appinventor.mit.edu/explore/
- Cargo-Bot—logic iPad game
  https://itunes.apple.com/us/app/cargo-bot/id519690804?mt=8
- Hopscotch (for up to intermediate–more complicated than Kodable)
  https://itunes.apple.com/ca/app/hopscotch-coding-for-kids/id617098629?mt=8

- Kodable
  https://www.kodable.com/
- Tynker
  https://www.tynker.com/

_____Here are more coding/programming apps that might be exactly what you're looking for (http://bit.ly/1MW58OA).

_____For an in-depth discussion on three iPad programming apps, read the article at the end of the Lesson, ***Want to Code on an IPad? Here are 3 Great Apps***.

Sequencing

_____Create a series of sequenced activities using slides that students must rearrange. These can be simple or involved, and might tie into inquiry taking place in the classroom. For example, enter life cycle slides into a slideshow tool like PowerPoint or Google Slides, share with students, and have them arrange in the correct order. In *Figures 10a-c*, what's the correct order for the slides?

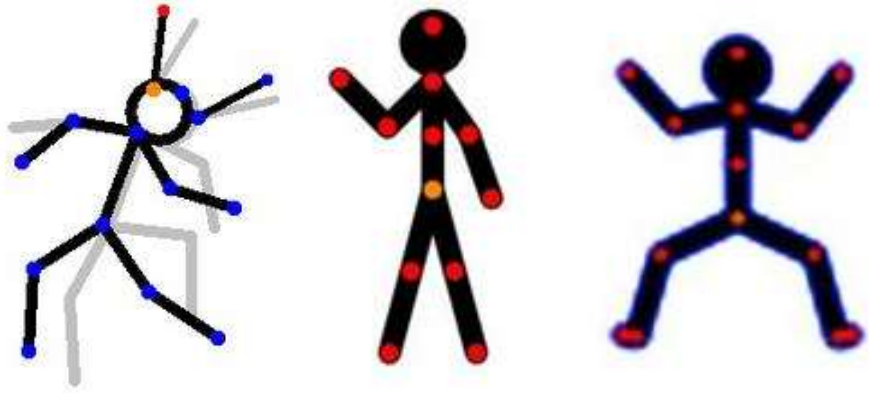
_____This can be done as a class or in small groups.

Figure 3a-c: Correct sequence

Animation

_____**Use a free program like** Pivot Stick Figure Animator **(**http://bit.ly/1IwbAXt**)** to program a stick figure. It's simple to use and students love exploring the possibilities of making their own creative animated stories.

Figure 4a-c—Stick figure animation



_____In *Figures 11b-c,* the red dots show places where the figure can be bent. Each frame, students

make a minor adjustment to the figure. By the time they finish and play the animation, the figure appears to be moving.

_____Pivot Stick Figure is a downloaded program. If you have IPads, try Stick Nodes (https://apple.co/1fB6D7H) or Scribble Movie (http://apple.co/1MSxBD1). If you use Chromebooks, try one of the other Coding options.

_____Save a screenshot (explain why this will not save the animation) to digital portfolios with student last name and project name. Here are options, depending upon your digital device:

- ***Windows****: Snipping Tool*
- ***Chromebook:*** *hold down control key and press window switcher key*
- ***Mac****: Command Shift 3 for a full screenshot; Command Shift 4 for partial*
- ***Surface tablet****: hold down volume and Windows button*
- ***iPad****: hold Home button and power button at same time*
- ***Online****: a screenshot tool like Jing, Nimbus, or Snagit*

_____Throughout class, check for understanding.

Class exit ticket: ***None***

Third Grade—Three Activities

Vocabulary	Problem solving	Skills
• *cells* • *coding* • *debug* • *Hour of code* • *pixel art* • *programming* • *sequence* • *symbolism*	• *I don't know how to use the programming tool (experiment; be a risk-taker)* • *I don't like coding (why?)* • *My partner does lots of the work (that's OK if you do your part)* • *I couldn't debug my program (start at the beginning)*	**New** *Coding/programming* **Scaffolded** *Problem solving*
Academic Applications *Math, critical thinking, habits of the mind*	**Materials Required** *Coding links, memberships in onsite program (i.e., Code.org)*	**Standards** *CCSS Standards for Math. Practice* *NETS: 4b, 6c*

Essential Question

How do I use a program I've never seen before?

Big Idea

By thinking critically, I can create something new and useful.

Teacher Preparation

- Talk with grade-level team so you tie into inquiry.
- If you're a lab teacher, arrange with stakeholders to extend lesson to one hour and fifteen minutes, to accommodate the nation-wide Hour of Code.
- Integrate domain-specific tech vocabulary into lesson.
- If you offer afterschool tech help and it's manned by students, verify they will be there.

Assessment Strategies

- *Anecdotal*
- *Completed exit ticket*
- *Worked well with partner*
- *Completed one hour of coding*
- *Joined classroom conversations*
- *Higher order thinking: analysis, evaluation, synthesis*
- *Habits of mind observed*

Steps

Time required: ***15 minutes to discuss problem solving, critical thinking, coding. 60 minutes to pursue hands-on coding (1 hour and 15 minutes preferred).***

Class warm-up: ***None***

_____Because this is one hour devoted to coding, skip presentations and Evidence Board.

_____Discuss critical thinking and problem solving. Does this apply to, say, Minecraft?

_____The reasons educators embrace coding are simple: **It teaches children to think.** Discuss fundamental programming concepts:

- *abstraction and symbolism–variables are common in math, but also in a student's education. Tools, toolbars, images–these all represent something bigger. See this post on the [symbolism of the word 'Turkey' (http://wp.me/pZUgb-2wS).](http://wp.me/pZUgb-2wS)*
- *creativity–think outside the box; develop solutions no one else has*
- *debugging–write-edit-rewrite; problem-solve; when you make a mistake, you don't throw your hands into the air or call for an expert. You look at what happened step by step and fix where it went wrong. Students should do the same.*
- *if-then thinking–actions have consequences*
- *logic–go through a problem from A to Z, understand the predictability of movements*

- *sequencing–know what happens when; mentioned in CCSS for grades 1 through 5*

_____Share this with students and get their thoughts:

"In 1997, the New York Times reported, 'It may be a hundred years before a computer beats humans at Go.' It took 16 years."

_____December will host the **Hour of Code**, a one hour introduction to coding, programming, and why students should love it. It's designed to demystify code and show that anyone can learn to be a maker, a creator, and an innovator.

_____This unit may be done individually or in small groups. It may be done any time during the school year.

_____Most students think programming looks like *Figure 12a* when it actually looks like *Figure 12b*:

Figure 5a-b—Coding

_____Do students remember coding activities from previous years—*Figures 13a-d:*

Figure 6a-d—Coding in K-2

_____This lesson presents three approaches. Pick one that works for your student group:

- *Hour of Code lessons*
- *Coding Pixel Art*
- *Miscellaneous coding websites*

Follow one of the free online Hour of Code programs

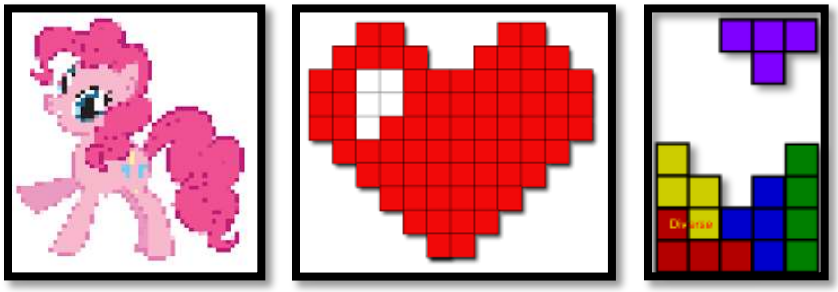
_____Websites like Code.org (Google for web address) offer full lesson plans for Hour of Code. This is the easiest way to get involved in programming as they do all the planning for you. This may be exactly what you need.

_____Before starting, review digital citizenship–especially privacy.

Coding Pixel Art

_____Pixel Art is the blocky drawing done, ala Minecraft. *Figures 14a-c* are examples:

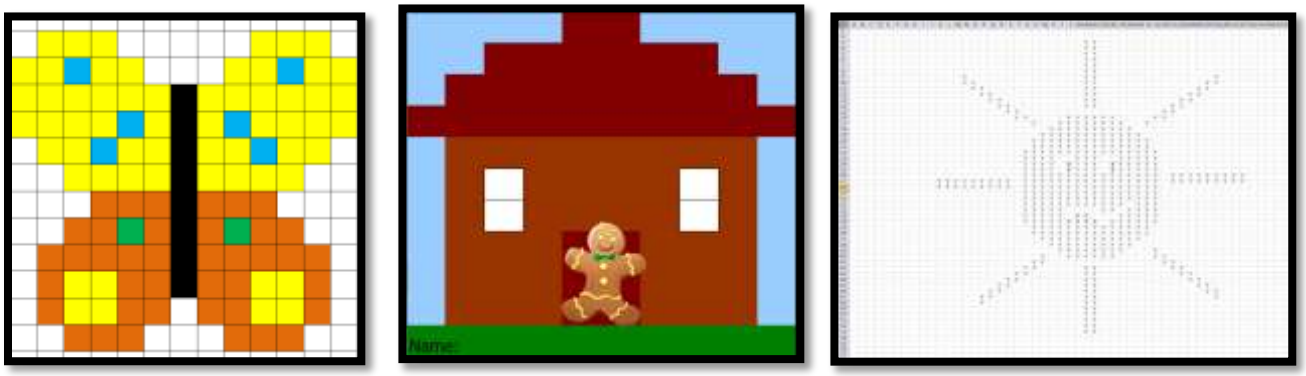
Figure 7a-c Pixel art



- *stair-step edges*
- *limited colors*
- *usually a focal centerpiece*

_____Gamers used pixelated art long before Minecraft in the popular Tetris (*Figure 14c*).
_____If you follow the Structured Learning Technology Curriculum, students have made pixel art in kindergarten and 1st grade (*Figures 15a-c*):

Figure 8a-c—K/1 Spreadsheet drawings



Next pages intentionally deleted

7th Grade: Scratch, AutoHotkeys, Wolfram/Alpha

Vocabulary	Problem solving	Homework
• *Background* • *Blocks* • *Broadcast* • *Control* • *Costume* • *Debug* • *Hotkeys* • *Motion* • *Operators* • *Remix* • *Script* • *Sequence* • *Sprite* • *Stage* • *Variables* • *Widget* • *Wolfram-Alpha*	• *I can't understand how to *** (Check resources, Help files, neighbors before asking teacher)* • *I can't remember how I *** (check scripts where you did this before)* • *I don't understand how to use a tool (right click and select 'help')* • *How do I know where scripts are (experiment)* • *How do I do basic skills (try Scratch Task Cards)* • *Is Scratch a drawing program or a presentation tool?* • *Can I use someone's script (that's 'remixing'—Scratch encourages it)* • *I just don't get it (see if you can try another lesson option)*	*Preview programming tool students will use in this lesson* *Add a blog post about the coding activity student would like to try. Include evidence.* *Review preparatory material*
Academic Applications *Math, critical thinking, problem solving*	**Skills Required** *Familiarity with problem solving, digital citizenship, keyboarding, programming*	**Standards** *CCSS: Math.Practice.MP* *NETS: 3b, 4b*

Essential Question

How can math be creative and collaborative?

Big Idea

I can learn mathematical ideas while thinking creatively

Teacher Preparation/Materials Required

- Have scratch program on digital devices.
- Have lesson materials online to preview.
- Ensure required links are on student digital devices.
- Ask what tech problems students had difficulty with.
- Integrate domain-specific vocabulary into lesson.
- Go to ScratchEd for tutorials, rubrics, assessments and more: http://scratched.media.mit.edu/.
- Talk with subject teachers about inquiry Scratch can support.

Assessment Strategies

- *Created sprite*
- *Completed project*
- *Posted blog article about Scratch (with screenshot) and commented on classmate's*
- *Completed warm-up, exit ticket*
- *Joined classroom conversations*
- *[tried to] solve own problems*
- *Decisions followed class rules*
- *Left room as s/he found it*
- *Higher order thinking: analysis, evaluation, synthesis*
- *Habits of mind observed*

Steps

Time required: ***360 minutes***
Class warm-up: ***Keyboarding on the class typing program, paying attention to posture***

_____**Homework listed on this lesson will be assigned the week before you start this unit—so students are prepared for the flipped classroom.**

_____Any questions from preparatory homework? Expect students to review upcoming unit and come to class with questions.

_____'Programming' is the buzzword among middle school students. They either want to do it, or are afraid of it. What does it mean? Who has their own website or blog? Who wants to write programs and/or apps? If they tried, what did they use? Discuss how these activities promote problem-solving, critical thinking, and computational thought.

_____Most people—students and adults—think programming looks like *Figure 27a* when it actually looks like *Figure 27b*:

Figure 9a-b--What programming feels like vs. what it is

_____Do students remember coding activities from previous years (*Figures 28a-e*)?

Figure 10a-e—Coding from previous years

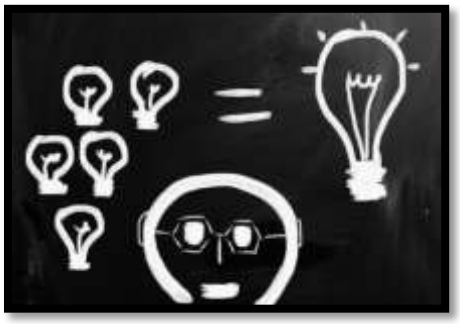
_____December will host Hour of Code (http://code.org), a one hour introduction to programming and why students should love it. It's designed to demystify "code" and show that anyone can learn to be a maker, a creator, and an innovator.

_____What is Scratch (https://scratch.mit.edu/): *A free download from MIT designed to teach pre-high school students programming basics without the techie-ness. With it, students create interactive stories, animations, games, and/or music.*

_____Whether you're a Common Core school or not, these eight constructs from Standards for Mathematical Practice regarding critical thinking tie flawlessly into Scratch programming:

- *Make sense of problems and persevere in solving them—Students must understand where they made a programming error and fix it.*

- *Reason abstractly and quantitatively—Visualizing coding requires an abstract understanding of what is occurring.*
- *Construct viable arguments and critique the reasoning of others—Coding and remixing requires students critique others' work.*
- *Model with mathematics—Translate available scripts to student needs, not unlike decoding a formula in mathematics.*
- *Use appropriate tools strategically—Coding offers a plethora of tools. The trick is to adapt strategically to student needs.*
- *Attend to precision—To get scripts to do what students want requires precision*
- *Look for and make use of structure—look at available tools, scripts, blocks, options, in selecting those which facilitate student needs*
- *Look for and express regularity in repeated reasoning—notice when a formula/program/script accomplishes goals.*



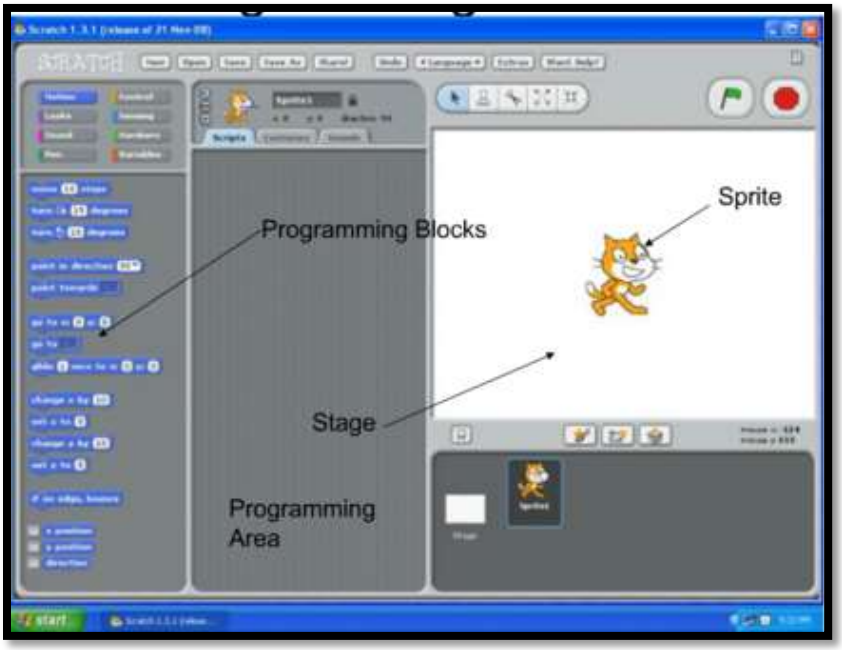
_____This lesson has three activities:

- *Scratch*
- *Auto Hotkeys*
- *Wolfram/Alpha widgets*

Scratch

_____Students work in groups. This is a self-paced student-directed unit. Provide a quick overview. In fact, after your screen tour, students will know 90% of what is required to complete the project. As you present, encourage students to listen for the following:

Figure 11—Scratch program page



- *What is background and how is it edited?*
- *What is broadcasting?*
- *How does one build/edit a sprite, make it glide?*
- *How does one add dialogue and recordings?*
- *How does a sprite move forward/backward and/or flip?*
- *How does one automate movement?*
- *How does one wait (under control)?*

Figure 12a-b—Scratch script and result

_____Open Scratch on class screen. Point out:

- *top toolbar with tools to save/share projects*
- *toolbar above stage where students duplicate/delete/grow/shrink their Sprite*
- *small stage, full stage, presentation mode tools*
- *how to connect and activate scripts*
- *three ways to create a Sprite and add costumes*
- *blocks—scripts that change with options*
- *control options*
- *green flag to automate scripts*
- *programming categories (motion, looks, sound, etc.)—demonstrate each*
- *drop-down menus available on some blocks/scripts*
- *tabs for sprites/backgrounds that change depending upon which you're in*

_____Take questions, but remember: You aren't teaching. You're introducing. Students are explorers and risk-takers in this project.

_____Provide a list of resources to help students find answers, like these (click the link and scroll down to Scratch): http://bit.ly/1knJOax.

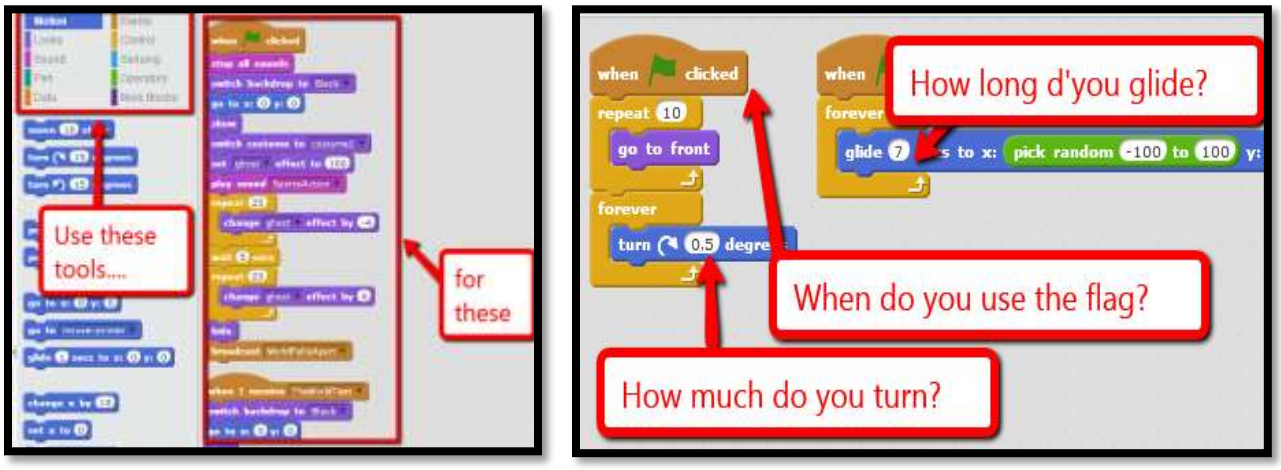
_____Before you help, students must try to solve their own problem. Here are strategies:

- *check resource list*
- *check Scratch website Task cards*
- *right-click on a tool and select 'help'*
- *check with a neighbor*
- *check Help (with Scratch's website)*

Figure 13—Scratch tools I

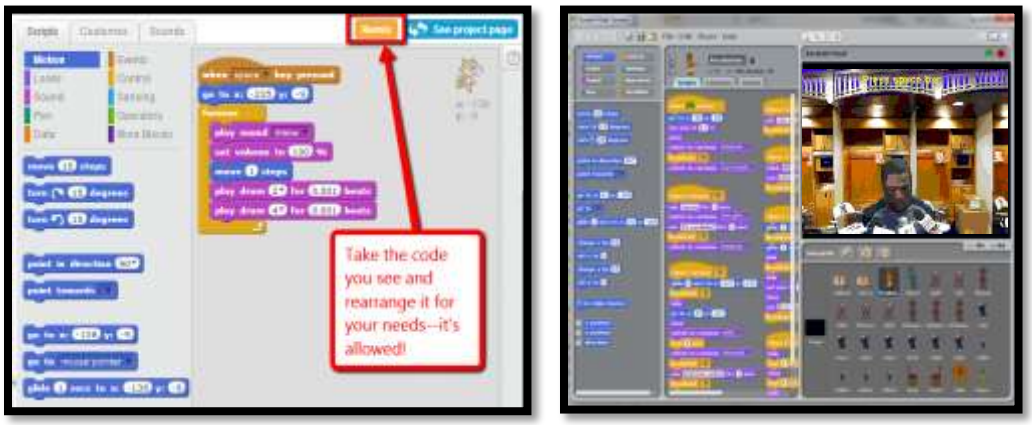
_____Give students time to view resource list and Task cards, experiment with tools, explore functions before beginning project.

Figure 14—Scratch tools II



_____When students have practiced skills, have them create an account on Scratch and download a project from another 7th grader (at http://scratch.mit.edu/latest/shared, search '7th grade'). Find a topic similar to one they will create. Explore how this student accomplished tasks; remix to suit project needs, then save remix to student portfolio (*Figures 33a-b*).

Figure 15a-b—Scratch remix

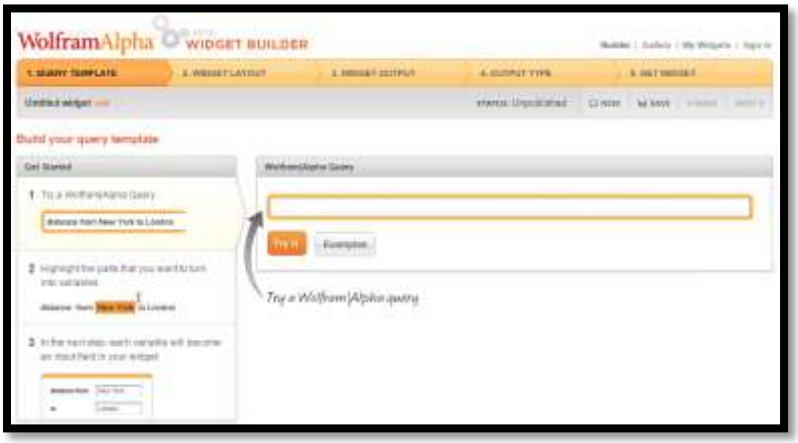


Next pages intentionally deleted

Code a Widget with Wolfram/Alpha

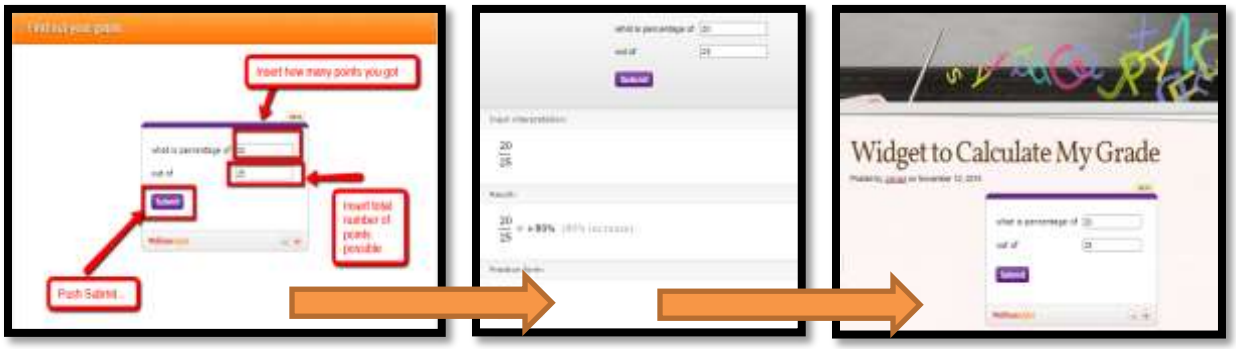
_____Widgets are free, personalized mini-apps that can do almost anything the user can program, from calculating the calories in a recipe to solving complex problems. Students can browse Wolfram/Alpha's gallery for a widget that fits their need and embed the code into their personal website, or build their own widget from scratch using the Builder tool. The level of difficulty will determine how long it takes so start simple during Hour of Code.

Figure 16—Wolfram/Alpha widget



_____Here's an example students can easily create to determine their grade (*Figures 39a-b*):

Figure 17a-b—Wolfram/Alpha completed widget; 39c—embedded in blog



_____Using the Wolfram Alpha embed code, add this to the student blog (*Figure 39c*):

Next pages intentionally deleted

Next pages intentionally deleted

_____Unit is student directed. Expect them to learn by exploring, sharing knowledge. Show this video of 2nd grader teaching SketchUp to classmates (http://www.youtube.com/watch?v=lahdM5v1fsw).

_____If you'd like: Have students go through this twenty-eight video Getting Started series: http://www.sketchup.com/intl/en/training/videos/new_to_gsu.html.

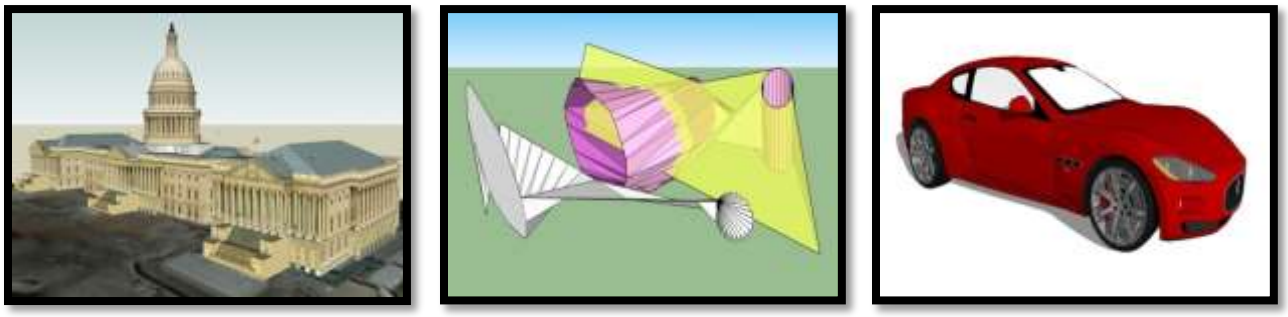
_____**Alternatively**, try *How-to SketchUp* (*http://www.aidanchopra.com/web-content*).

_____Note: Video links change. You can find your own resources by searching YouTube, SchoolTube, Woopid, or similar (Google for addresses).

_____Open SketchUp. Students browse online documentation and videos. Encourage them to think back to the videos they watched for homework and in class when they have a question. Replay those videos as needed to be self-directed and self-motivated in this lesson.

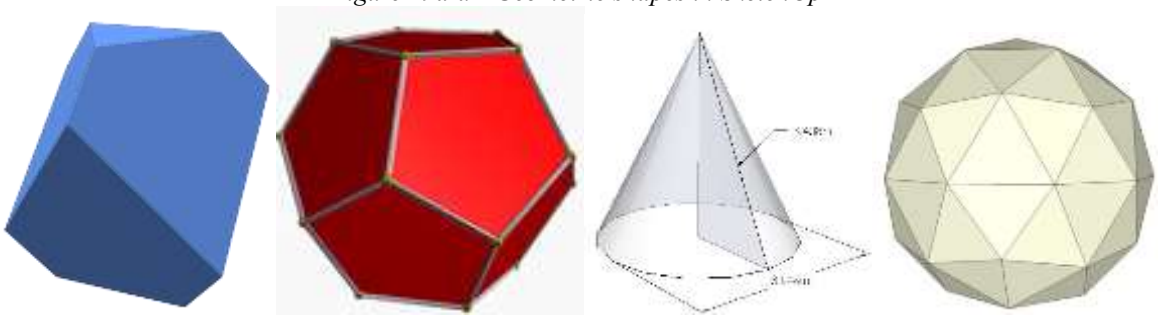
_____Introduce the SketchUp Warehouse. Browse to see what has been created (see *Figures 50a-c*).

Figure 18a-c—Designs from SketchUp Warehouse



_____Next: In groups, create several 3D geometric shapes like *Figures 51a-d* (from SketchUp Warehouse):

Figure 19a-d—Geometric shapes in SketchUp



_____Next: In groups, create a building to scale (*Figure 52*):

Figure 20—House in SketchUp

_____Start by watching this video: [http://bit.ly/1YmNsQy.](http://bit.ly/1YmNsQy)
_____Next: Students will complete one of the following tasks that integrate SketchUp with math, geography and science:

- *Find a SketchUp of a real building in the warehouse. Try to reproduce it. Then, compare your design to pros. In Figures 53a-b, which is real and which is SketchUp:*

Figure 21a-b: Which is real? Which is SketchUp?

- *Create an icosahedron (see https://youtu.be/Kz3OpsgDiz0) like Fig. 54a:*
- *Design a building on your campus and upload to Google Earth (or Warehouse). Do you recognize Figure 54b—Eliot School in St. Louis Missouri (from warehouse)?*

Next pages intentionally deleted

More Themed Bundles of Lessons from Structured Learning

- Art
- Coding
- Common Core
- Geography
- Google Earth
- History
- Inquiry
- Language Arts
- Math
- Problem Solving
- Writing