ScratchProgramming.org:
An Educator's Guide to Scratch Programming

Table of Contents

Introduction

About

This guide was created by Julian Screawn. It was created in conjunction with my master's project, which is a guide to Scratch programming for educators. The purpose of the guide is to enable Scratch educators:

- To create environments where students can have opportunities to develop Scratch usage and programming skills.
- To explore the ways in which Scratch can be used as a tool to enhance the teaching-learning process across the curriculum.

Scratch supports the development of 21st century learning skills such as critical thinking, problem solving, communication, collaboration, creativity and innovation.

The guide will be targeted at teachers (Grade 3 and up) who wish to use Scratch as a tool for helping students develop these 21st century skills. It is hoped that the guide will be helpful to technology teachers and subject teachers who wish to expand their tools for teaching and integrating technology.

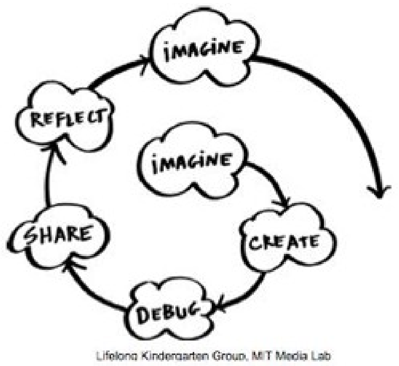
Content for the guide is based on both research and my own personal experiences as a Scratch educator.

Underlying Philosophy

One of the main goals of the Scratch program designers was to facilitate learn by designing.

Learning by design :

- Gives students greater sense of control and responsibility for the learning process.
- Encourages creative problem-solving.
- Allows for the designing of projects that are interdisciplinary(art, technology, math, and sciences).
- Helps kids learn to put themselves in the minds of others, since they need to consider how others will use the things they create.
- Provides opportunities for reflection and collaboration.
- Sets up a positive-feedback loop of learning, where students can build on ideas.



(Resnick,n.d.)

This approach to learning and teaching is inspired by the constructivist and constructionist theories of learning and education. Most activities recommended in this guide are based on the constructivist approach to learning.

Why use Scratch ?

1. Supports the development of 21st century learning skills. According to the Scratch developers, Scratch supports the nine types of 21st century learning skills identified by the Partnership for the 21st Century (http://www.p21.org) ;these skills include: thinking creatively, communicating clearly, analyzing systematically, collaborating effectively, designing iteratively, and learning continuously.
(Rusk,Resnick,& Maloney,n.d.).

2. Supports the development of programming skills by making programming more engaging and accessible for children, teens, and others. According to the National Research Council(NRC)(1999), algorithmic thinking and programming is a 21st century skill to be learned by all students.

3. It's a tool used for teaching and learning across the curriculum. According to Crook, (2009) Scratch offers the teacher an opportunity to embed the computer into everyday school activities by getting the class to develop skills in digital literacy related to a variety of curriculum topics.

4. It's free. Sixty-three percent of the teachers surveyed by PBS LearningMedia (2012) stated that limited budget for technology adoption was the biggest barrier to accessing technology in the classroom.

What can I do with Scratch ?

According to the Scratch homepage,

> Scratch is a programming language that makes it easy for users to create their own interactive stories, animations, games, music, and art -- and share their creations on the web.

When students design projects with Scratch they develop fluency with digital technology using the skills that will be needed for the 21st century.

Examples:

- Simulations: demonstrate concepts by imitating something that is done in reality. For example a probability simulation, such as a coin or dice toss experiment. See video http://www.scratchprogramming.org/video.php?vid=23 for a probability demonstration.
- Multimedia: create interactive puzzles, quizzes, demonstrations and class presentations. See video http://www.scratchprogramming.org/video.php?vid=24 for a quiz demonstration.
- Music: create interactive instruments, music videos, or games that prompt students to play notes in the correct order. See video http://www.scratchprogramming.org/video.php?vid=25 for a saxophone demonstration.
- Art: create interactive and non-interactive art projects. See video http://www.scratchprogramming.org/video.php?vid=26 for a Math Art demonstration.

- Storytelling and Journals: create interactive stories or animations to support students narrative and creative writing skills. See video http://www.scratchprogramming.org/video.php?vid=27 for an interactive story demonstration.

- Role play:role play real world professions, for example pretending to be a game designer and design a new game.

Getting Started

Start Here

New to Scratch

Experienced User

From the Scratch website:

1. Download Scratch: http://info.scratch.mit.edu/Scratch_1.4

2. Read the Scratch introductory guide: http://info.scratch.mit.edu/sites/infoscratch.media.mit.edu/docs/ScratchGettingStartedv14.pdf

3. See Scratch introduction videos: http://info.scratch.mit.edu/Video_Tutorials

4. All videos and activities that were created for this guide are associated with a dodge ball game that was created in Scratch. It is **recommended** that users spend some time **reviewing the dodge ball game** before reading through this website guide. To download the dodge ball game: http://www.scratchprogramming.org/documents/Dodgeball.sb

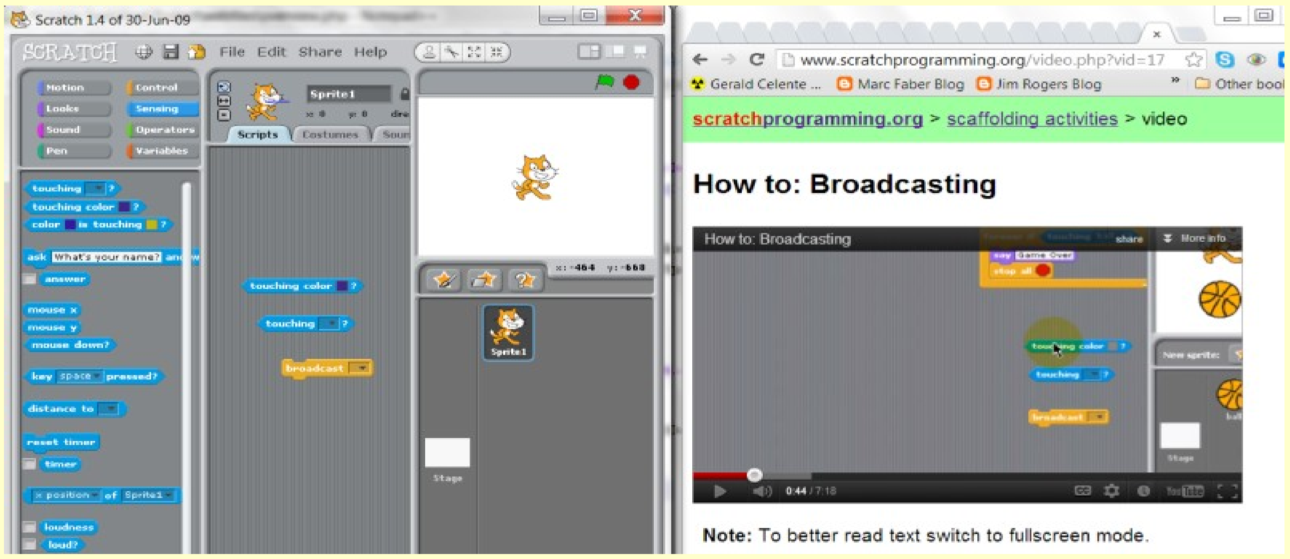
Tip: If you wish to have Scratch open while viewing the scaffolding "How to" videos on this site then split your screen between Scratch and your web browser.

How to split your screen:

Mac users: http://www.ehow.com/how_8599299_split-screens-macbook.html,

Windows users: http://www.ehow.com/how_7260916_split-screen-pc-monitor.html.

The screen can also be resized manually.



Note: To better read text switch to fullscreen mode.

Basic Skills

Scaffolding Activities

> Like training wheels computer scaffolding enables learners to do more advanced activities and to engage in more advanced thinking and problem solving than they could without such help. (NRC, 2000, p.214)

One of the best ways to introduce Scratch is to give students a set of fun challenges that scaffold their learning of basic concepts and skills.

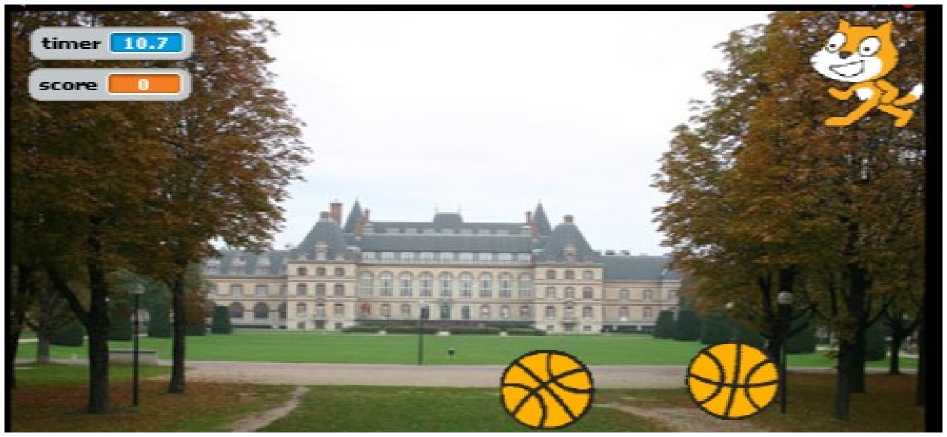
> According to (Alber,2011) "Scaffolding is breaking up the learning into chunks and then providing a tool, or structure, with each chunk" (par. 2).

How to scaffold learning with Scratch:

- Start with an interesting level appropriate Scratch game,animation or project and break it up into chunks(challenges or explorations).
- Provide support (teacher does student watches/helps) and a challenge (student does teacher watches/helps) for each chunk.
- Create objectives for each chunk.
- Challenges can be completed individually, in pairs or groups.

For example:

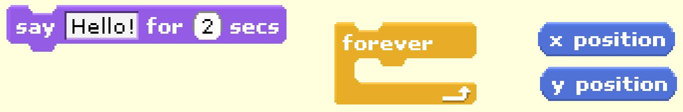
Students will create the dodge ball game below with Scratch.

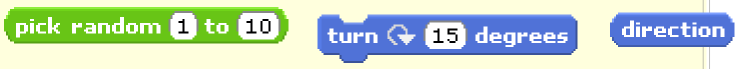
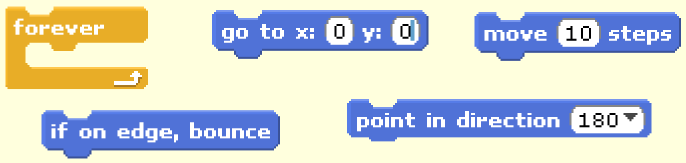


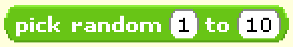
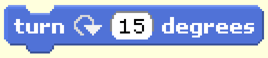
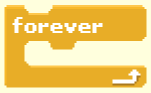
Notes:

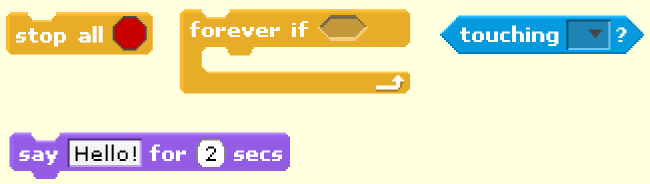
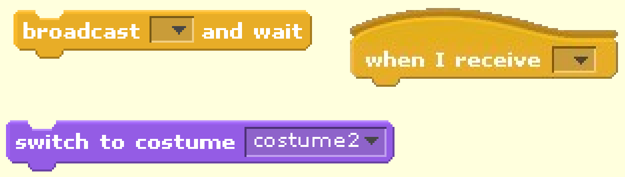
- Each challenge should be designed to introduce a new skill or concept.
- Challenges should be sequenced from the easy to more difficult in a way where they build on each other to complete a project (game, animation, story, etc.).
- Challenges don't always necessarily need to be done in order.
- Solutions to challenges may differ.

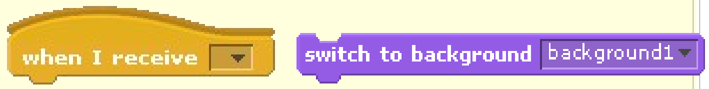
The following table breaks the dodge ball game up into learning chunks:

Chunk	Support	Challenge
Screen Position	Introduce: • Position on the screen. • Position variables. • Xy coordinate	Get the cat to say its screen address(position) using the following blocks:  say Hello! for 2 secs forever x position y position

	system. • Directed numbers. How to video: http://www.scratchprogramming.org/video.php?vid=1	Videos: • Challenge: http://www.scratchprogramming.org/video.php?vid=6 • Solution: http://www.scratchprogramming.org/video.php?vid=7
Direction	Introduce: • Sprite Direction. • Measurement of angles. • Outcomes of the random data process in Scratch. How to video: http://www.scratchprogramming.org/video.php?vid=2	Get the cat to say its current direction and position. The direction should be random. Add the following blocks to the screen position solution:  pick random 1 to 10 turn ↻ 15 degrees direction Videos: • Challenge: http://www.scratchprogramming.org/video.php?vid=8 • Solution: http://www.scratchprogramming.org/video.php?vid=9
Movement	Introduce: • The three motion blocks: go to, glide and move.	Get a ball to start at the middle top of the screen and fall to the bottom, and then bounce back up again. Use these blocks:  forever go to x: 0 y: 0 move 10 steps if on edge, bounce point in direction 180

	How to video: http://www.scratchprogramming.org/video.php?vid=3	Videos: • Challenge: http://www.scratchprogramming.org/video.php?vid=4 • Solution: http://www.scratchprogramming.org/video.php?vid=5
Random Movement	No support required.	Get the ball to move unpredictably around the screen. Add these blocks:  pick random 1 to 10  turn ↻ 15 degrees Videos: • Challenge: http://www.scratchprogramming.org/video.php?vid=10 • Solution: http://www.scratchprogramming.org/video.php?vid=11
Following the mouse cursor	No support required.	Have the cat sprite follow the mouse cursor around the screen. Choose one block from Motion and add this block:  forever Videos: • Challenge: http://www.scratchprogramming.org/video.php?vid=14

		• Solution: http://www.scratchprogramming.org/video.php?vid=13
Sensing	Introduce: • The touching color block. How to video: http://www.scratchprogramming.org/video.php?vid=12	Get the game to stop and have the cat say "game over" when the ball touches the cat.Use these blocks:  stop all forever if touching ? say Hello! for 2 secs Videos: • Challenge: http://www.scratchprogramming.org/video.php?vid=16 • Solution: http://www.scratchprogramming.org/video.php?vid=15
Broadcasting	Introduce: • Broadcast block. • Broadcast and wait block. • The differences between the two broadcasts.	Broadcast a game over message when the cat gets hit by the ball. The cat should should receive the message that it sends, and then change to a new costume. Add these blocks to the cat script:  broadcast and wait when I receive switch to costume costume2 Videos:

	How to video: http://www.scratchprogramming.org/video.php?vid=17	• Challenge: http://www.scratchprogramming.org/video.php?vid=18 • Solution: http://www.scratchprogramming.org/video.php?vid=19
Broadcasting in Action	No Support required.	Broadcast a game over message when the cat gets hit by the ball. The stage should receive the message, and then change to a game over background. Add the following blocks to the stage script:  when I receive switch to background background1 Videos: • Challenge: http://www.scratchprogramming.org/video.php?vid=20 • Solution: http://www.scratchprogramming.org/video.php?vid=21

Creative Activities

Unlike many traditional programming languages, Scratch is relatively easy to pickup and learn by both students and teachers. The nature of Scratch is self learning. According to the creators of Scratch:

> "A key design goal of Scratch is to support self-directed learning through tinkering and collaboration with peers" (Maloney, Resnick, Rusk, Silverman, Eastmond, 2010, p. 1).

The design of Scratch enables tinkering and experimenting which often results in a few students quickly becoming Scratch experts; teachers can then utilize these experts as peer tutors.

Cultivating Creativity:

- In order to stimulate motivation, teachers should encourage students to experiment with Scratch tools and create projects (game, animation, art, etc.) of their own preference.
- Teachers should first introduce the tools of Scratch through scaffolding activities.
- Once students have some basics down they should be free to show off their creativity and take their projects further by tinkering and collaborating with others locally (in their classroom) or globally (Scratch learning-sharing community website).
- Take away the scaffolding as students become more able to problem-solve and create their own projects of preference.
- After the completion of scaffolding activities students should be provided with an opportunity to take their projects further. In order to get ideas flowing and to motivate students it is best to first brainstorm ideas.

Brainstorming Strategies:

- Divide students into pairs or small groups and get them to think about ways to further develop the game. Then collect and discuss ideas as a class or have each group present their ideas.
- Have students search the internet for ideas, a great starting point is the Scratch website (http://scratch.mit.edu/). Students can browse projects uploaded to the site to get new project ideas and learn new programming techniques.
- Set a brainstorming time limit.
- Encourage remixing or building on one another's ideas

For example:

On the scaffolding activities page a basic dodge ball game was created. Ideas for further developing the dodge ball game could include:

- Adding more balls for the cat to dodge.
- Adding a survival time feature.
- Add another level with a different background.
- Add sounds to the game.

Watch a video containing additions to the dodge ball game at:

http://www.scratchprogramming.org/video.php?vid=22

Facilitating Collaboration

The idea of creativity should not always be thought of as a single student thought process. Creativity also takes place in a social context.

> "An idea or product that deserves the label 'creative' arises from the synergy of many sources and not only from the mind of a single person. It is easier to enhance creativity by changing conditions in the environment than by trying to make people think more creatively" (Csikszentmihalyi, 1996, p. 1).

Environments that support Scratch collaboration:

The Scratch website: the Scratch designers emphasized sharing and collaboration when they created the Scratch community website; "the Scratch Online Community makes programming more engaging by turning it into a social activity" (Monroy-Hernandez & Resnick, 2008, p.50). On the site members can:

- Post projects and get project ideas from other uploaded projects.
- Download and remix other student projects.
- Form online design teams;that is work on projects with other members around the world.
- Offer and get help from other members through forums.
- Offer and receive feedback on projects and ideas.
- Rate projects and offer up challenges.

The classroom: a similar collaborative environment can be created in the classroom by:

- Providing students with project feedback strategies. For example giving students a project feedback handout that helps to guide them in giving feedback.
- Creating feedback teams.
- Having students share their projects on the school network or have them create multiple copies of their projects for sharing.
- Creating design teams for collaboration on projects.

For example:

> The Jigsaw collaborative or cooperative technique can be used to perform appropriate programming activities within Scratch. (Theodorou & Kordaki, 2010).

The Jigsaw technique is described in 10 easy steps at the Jigsaw.org website. Outlined below is an implementation of this technique using a Scratch project example. In the jigsaw groups, students will share knowledge and then work on the Scratch game dodge ball.

1. Divide students into jigsaw groups.
2. Appoint one student from each group as the leader.
3. Divide the project into segments, similar to what was done in the scaffolding activities. For example: Screen Position, Direction, Movement, Sensing, and Broadcasting.
4. Assign each student to learn one segment.

5. Give students time to research and tinker with their segment to become familiar with it.
6. Form temporary "expert groups" by having one student from each jigsaw group join other students assigned to the same segment. Give students in these expert groups time to discuss the main points of their segment and to rehearse the demonstrations they will make to their jigsaw group.
7. Bring the students back into their jigsaw groups.
8. Ask each student to present her or his segment to the group. Encourage others in the group to ask questions for clarification.
9. Float from group to group, observing the demonstrations. If any group is having trouble (e.g., a member is dominating or disruptive), make an appropriate intervention
10. At the end of the demonstrations, get students to work on the dodge ball game individually,in pairs or in small groups. Students can seek help from experts or the teacher as they work on their projects.

(Aronson, 2008)

Programming Skills

Problem Solving

The Scratch programming languages was designed for educational use, to support the constructionist approach to learning which encourages creative problem-solving. Students will be problem solving as soon as they load up Scratch.

Although, Scratch programming facilitates higher order thinking such as problem solving skills, teachers can provide instructional support to students, to help them think through difficult programming problems. This can involve having students create algorithms, that is the breaking down of problems into smaller sub-components, and exploring multiple solutions to problems.

Thinking through problems

Getting students to master the process of thinking through programming problems and determining the best method of solving each problem is made easier with Scratch. One of the advantages Scratch has over traditional programming languages is its ability to easily allow users to visualize the results of their programming (solutions to problems) on the screen.

Scratch simulates traditional programming, by providing learners with a simple visual drag-and-drop user interface. This visual nature of Scratch allows students to test different ideas or approaches to a problem and to more easily learn what works and what does not.

The results of a study on the effects of simulation games on the learning of computational problem solving demonstrated that simulation games are an effective approach to assisting novice programmers to learn computation problem solving skills; the study found that "simulation games based on Paperts' constructionism may improve problem solving" (Chen-Chung, Yuan-Bang, & Chia-Wen, 2011, p. 1916).

When applying Constructivist learning theory to problem solving within Scratch, students should:

- Create their own algorithms for solving Scratch programming problems. They should not to be taught one specific algorithm, for example long division.
- Be encouraged to discuss, reflect on, and demonstrate strategies for problem solving.
- Solve problems collaboratively.
- Problem solve in authentic contexts.

A strategy for problem solving with Scratch

1. Give students an opportunity to practice writing and developing their own algorithm (solutions) on paper first.

2. As a class discuss, demonstrate and reflect on different solutions.

3. Next have students develop a visualization of their solution with Scratch. This could be a simulation, a game, or an animation. This can be done in small groups.

For example:

How to make a cup of tea

1. In small groups ask students to write out on paper a set of instructions to describe how to make a cup of tea. Tell them that the computer needs to know in detail every step.

2. As a class, ask students to share their instructions and note any differences or omissions.

3. Discuss the problems involved in creating an algorithm (set of instructions)

4. Next have students develop a visualization of their solution to the problem in Scratch. This could be a simulation, presentation, a game, or an animation.

See a video example of a presentation solution to the cup of tea algorithm at:

http://www.scratchprogramming.org/video.php?vid=31

Although this is an easy example, it is important to stress the need for instructions to be precise. You can get students/groups to write algorithms for other students/groups to follow and test out. This activity will reinforce the need for precision in algorithms.

Exploring multiple solutions

Students should learn that there are many different ways to program games, simulations and animations. Students should be encouraged to explore different solutions; this will help strengthen both their problem solving and programming skills, and give them confidence in creating their own algorithms (solutions).

Activities that explore multiple solutions allow students to see other ways of doing things, enabling them to construct new meanings through the context of their own experience(Dabbagh, 2005). Moreover, the experiencing of different perspectives is necessary for the development of problem solving abilities, creativity and advanced mathematical thinking.(Leikin, Levav-Waynberg, Gurevich, & Mednikov, 2006).

Tips for facilitating multiple solutions

- Challenge students to discover multiple approaches and/or solutions to programming problems.
- Share student solutions.
- Have students download and explore similar projects/solutions from the Scratch website.

Learning from Projects

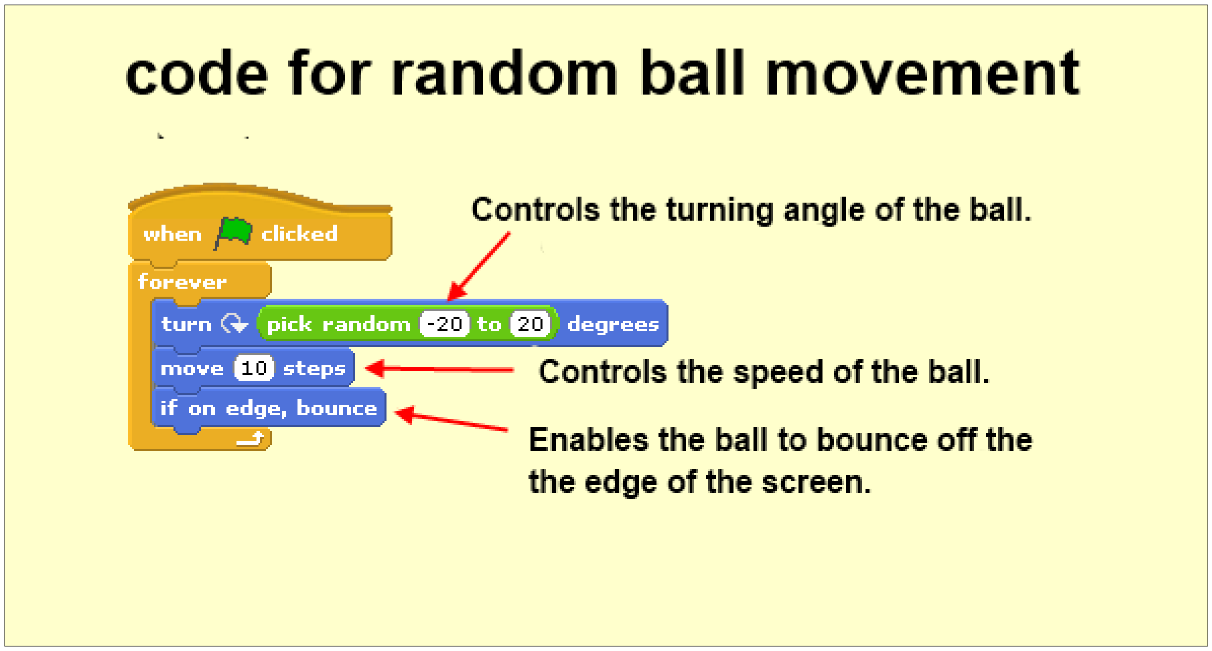
Experimenting with projects

Objective: enable students to better understand the role of programming constructs.

Students are asked to work with a completed project and experiment with specific blocks (programming constructs) from the code of the program. This experimentation could include changing the position of the blocks, or changing the value of some variables; it enables students to gain a better understanding of the roles of specific constructs. (Kordaki, 2012).

This exercise allows for the scaffolding of basic computer programming constructs. It is a good way to start the learning of programming constructs, as the exercise does not require students to build programs or algorithms.

For example:



```
when [green flag] clicked
forever
    turn ↻ (pick random (-20) to (20)) degrees
    move (10) steps
    if on edge, bounce
```

code for random ball movement
Controls the turning angle of the ball.
Controls the speed of the ball.
Enables the ball to bounce off the the edge of the screen.

In the above code example students can experiment with the code as follows:

- Enter different random angles to which the ball can turn. This helps students understand the concepts of direction and randomness.
- Change the number of steps the ball moves. This helps students better understand speed and movement.
- Remove the "if on edge, bounce" block and observe the changes.
- Replace the "Forever" control construct with a "Repeat" construct. Helps students to understand the differences between the constructs "Repeat" and "Forever".

Modifying projects

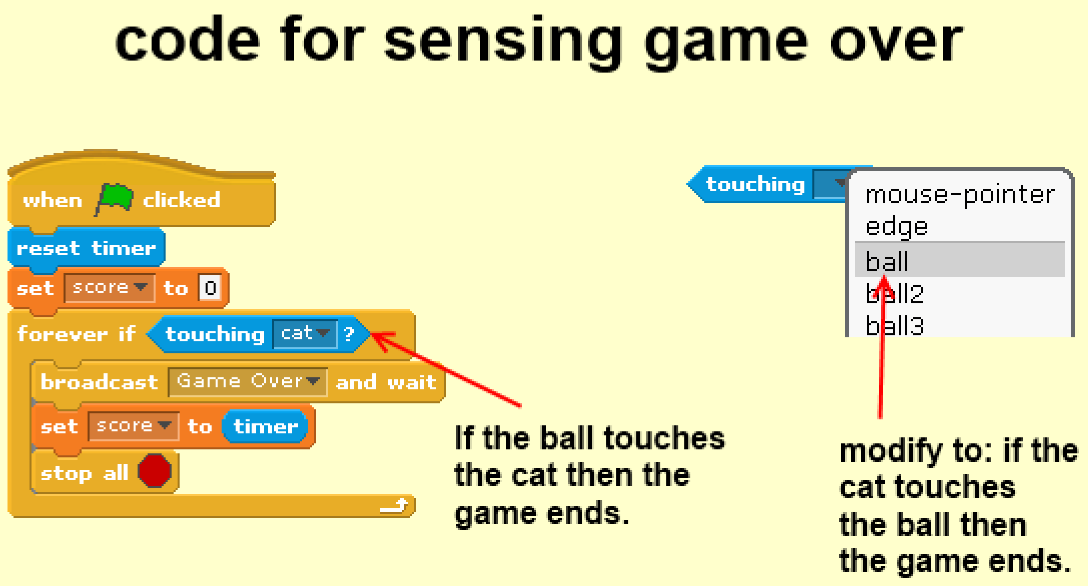
Objective: enable students to expand on, or use previously acquired programming knowledge to modify projects.

The idea is to have students use previously acquired knowledge to modify Scratch projects by producing a different result or output. The benefit of this activity is that students can be

> "sheltered by the context of the already working project in order to appropriately face the challenges of its modification" (Kordaki M.,2012, p.4).

For example:

Dodge ball game:



code for sensing game over

```
when [flag] clicked
reset timer
set score to 0
forever if touching cat ?
    broadcast Game Over and wait
    set score to timer
    stop all
```

If the ball touches the cat then the game ends.

modify to: if the cat touches the ball then the game ends.

In the above example students can modify the dodge ball game so that the game ends when the cat touches the ball as opposed to the ball touching the cat.

The above code modification enables students to:

- Build on, or apply their knowledge of sensing.
- Acquire a better understanding of the importance of coding for specific objects; that is students must move the game over script from the cat to the ball in order to make the modification work.
- Help students understand the concept of duplicate code; that is reducing code repetition. Students will realize that putting the game over script on the cat object as opposed to the ball will result in code repetition whenever a new ball is introduced.

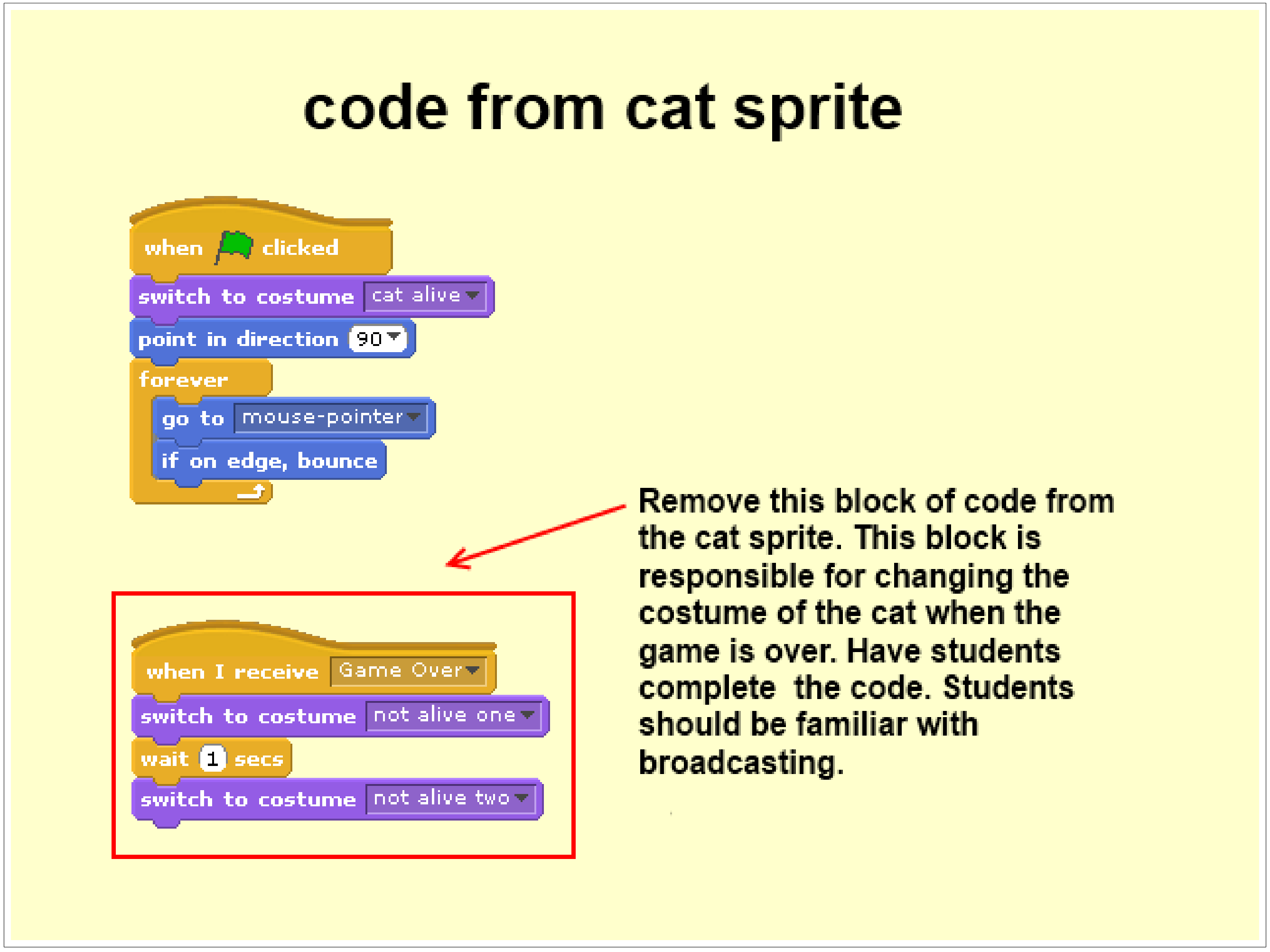
Coding Challenges

Scratch provides students with the chance to correct their programming attempts through trial and error in a visual environment; this makes it easier for learners to develop coding skills. The coding activities listed below can be used to challenge students and help them develop coding skills. Although these coding activities are normally created by the instructor they can also be created by the students to challenge each other.

Completing code

Objective: enable students to reflect on and apply Scratch programming knowledge and skills to complete incomplete code in a working project.

Take a working project and remove some of the code, then give students an opportunity to complete the project, that is fill in the missing code. Students should be able to see the output of the working project; students can use the working output along with the incomplete code as a guide to completing the code. (Kordaki, 2012).



code from cat sprite

```
when [flag] clicked
switch to costume [cat alive]
point in direction (90)
forever
    go to [mouse-pointer]
    if on edge, bounce
end

when I receive [Game Over]
switch to costume [not alive one]
wait (1) secs
switch to costume [not alive two]
```

Remove this block of code from the cat sprite. This block is responsible for changing the costume of the cat when the game is over. Have students complete the code. Students should be familiar with broadcasting.

For novice programmers, the programming blocks needed to complete the project and produce the correct output can be provided by the instructor. That is students are required only to assemble the blocks in the correct sequence. For a more difficult challenge the students must choose the correct blocks themselves to complete the code.

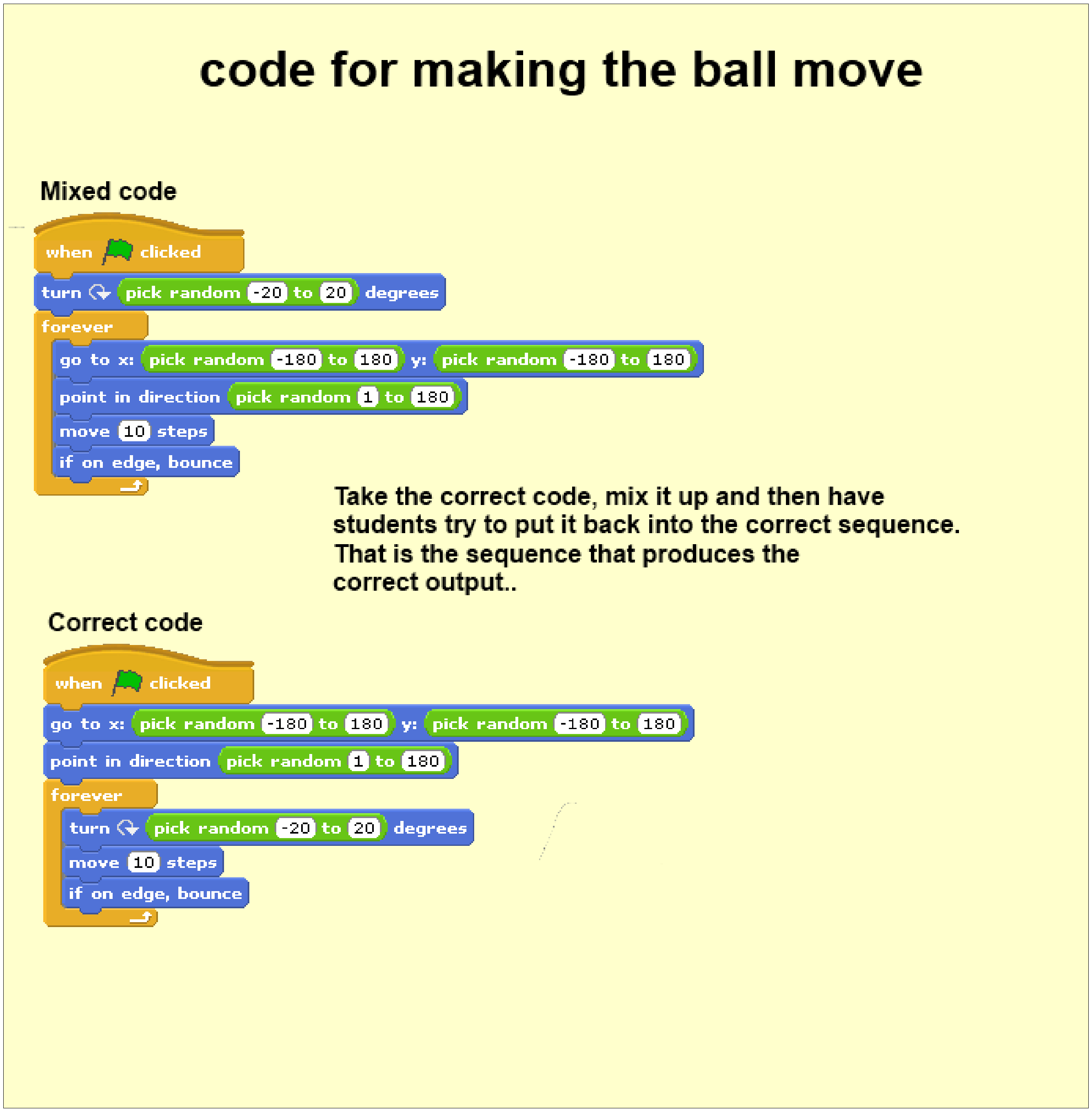
Mixing code

Objective: enable students to reflect on, and apply Scratch programming knowledge and skills to re-arrange mixed code.

Take a working project and mix-up some of the code, then give students an opportunity to arrange the code so that it produces the correct output. Students should be able to see the correct output on the screen beforehand.

This activity along with the completing code activity can be used to scaffold the learning of computer programming; these activities are easier because students do not have to develop the code themselves, they are given the blocks and only have to experiment until they find the appropriate sequence of commands that produces the correct output. (Kordaki, 2012).

For example:



code for making the ball move

Mixed code

```
when 🏁 clicked
turn ↻ pick random -20 to 20 degrees
forever
    go to x: pick random -180 to 180 y: pick random -180 to 180
    point in direction pick random 1 to 180
    move 10 steps
    if on edge, bounce
```

Take the correct code, mix it up and then have students try to put it back into the correct sequence. That is the sequence that produces the correct output..

Correct code

```
when 🏁 clicked
go to x: pick random -180 to 180 y: pick random -180 to 180
point in direction pick random 1 to 180
forever
    turn ↻ pick random -20 to 20 degrees
    move 10 steps
    if on edge, bounce
```

Correcting code

Objective: enable students to develop an awareness of programming errors commonly made in Scratch by novice programmers.

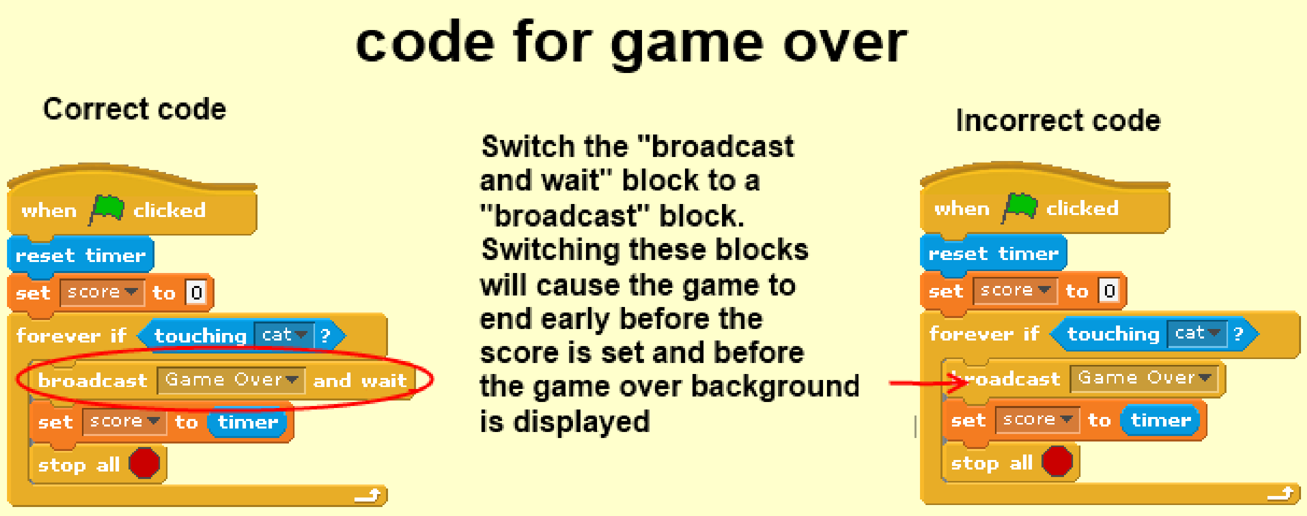
Students should be given a block of code that contains an error; that is producing incorrect output. This error should be reflective of a common error that is usually made by novice Scratch programmers.

Demonstrate to students the correct output and then ask them to correct the error; this activity is more difficult as students have to find specific mistakes included in the given code and also to make corrections so that it produces correct output. (Kordaki, 2012).

It is best to start with simple errors and work to more difficult challenges as students become more proficient.

For example:
Common error: Students often get confused with the differences between the "broadcast" and "broadcast and wait" blocks.

Dodge ball game:



```
when [green flag] clicked
reset timer
set score to 0
forever if touching cat ?
    broadcast Game Over and wait
    set score to timer
    stop all
```

Switch the "broadcast and wait" block to a "broadcast" block. Switching these blocks will cause the game to end early before the score is set and before the game over background is displayed

```
when [green flag] clicked
reset timer
set score to 0
forever if touching cat ?
    broadcast Game Over
    set score to timer
    stop all
```

Advanced Activities

The advanced activities listed below are important as they help to develop advanced thinking skills. These skills, such as problem-solving and decision making are important for both the personal and professional life of students. (Dabbagh, 2005).

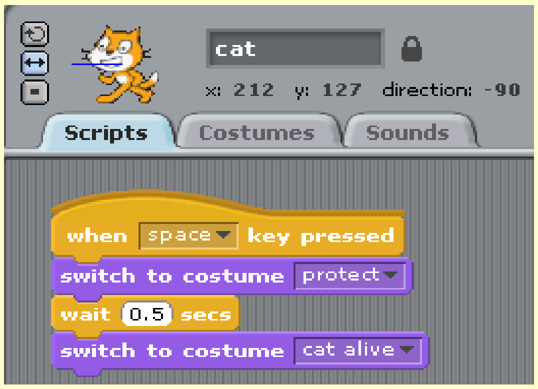
Predicting output

Objective: enable students to synthesize all Scratch programming knowledge and skills to predict programming outcomes.

Making predictions is a difficult task, it requires that students have reached an operative level of development (Piaget) and have an understanding of all the Scratch programming constructs (Kordaki,2012). In order to make predictions students should be able to use abstract thinking for solving problems and have the ability to imagine the outcome of particular actions. For example:

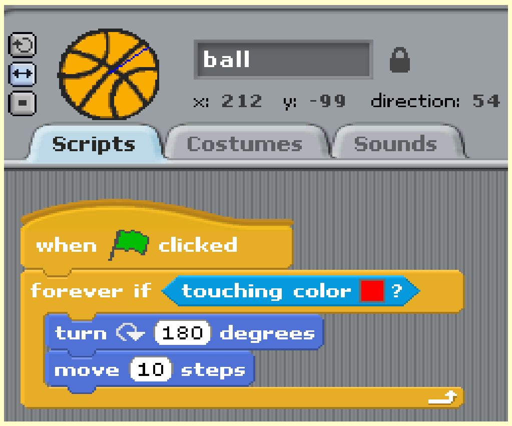
Dodge ball game:

code added to the cat and ball scripts



```
cat
x: 212  y: 127  direction: -90
Scripts  Costumes  Sounds

when [space] key pressed
switch to costume [protect]
wait (0.5) secs
switch to costume [cat alive]
```



```
ball
x: 212  y: -99  direction: 54
Scripts  Costumes  Sounds

when [flag] clicked
forever if <touching color [🟥]?>
    turn (180) degrees
    move (10) steps
```

Get students to predict what will happen if the above scripts are added to the game.

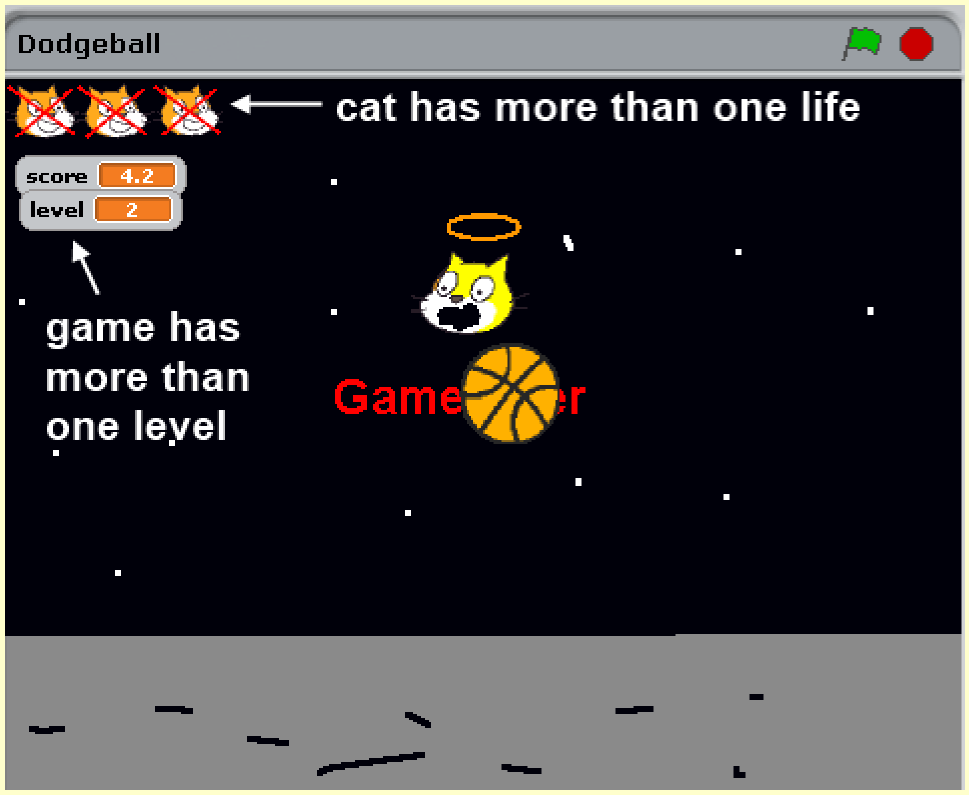
Black-box activities

Objective: enable students to synthesize all Scratch programming knowledge and skills to formulate code for a particular outcome.

Students are asked to develop the code for a particular output as it runs in the Scratch output window. Like predicting outcome this activity requires higher level skills, like "thinking skills such as reversible thinking, analytical and synthetic thinking, as well as reflection, prediction, hypothesis generation and exploration" (Kordaki, 2012, p.4).

For example: The output for the dodge ball game now has two additional components, both extra lives and levels. Looking at the output students should be able to produce the code.

Dodge ball game:



Curriculum Integration

Technology Integration

Integrating technology into the curriculum is more than just learning basic computer skills, how to use the internet, or how to use software programs. Students need to use technology for accomplishing goals and solving real world problems in a way that is similar to how these skills are being used in a real world setting.

> "Technology integration means viewing technology as an instructional tool for delivering subject matter in the curriculum already in place." (Woodbridge, 2004., par. 3)

Students become active learners and develop their problem solving, critical-thinking, and creativity skills when creating Scratch projects. Teachers must move from traditional teaching methods to engaging students in problem-based or project based learning which is student-centered.

For example:

In the table below is a model for Scratch Integration; it is an application of the The Apple Classrooms of Tomorrow (ACOT) stages of technology integration. These stages help teachers to integrate technology into teaching and learning.

Stage	Examples of what teachers do with Scratch
Entry	Learn the basics of using Scratch. For teachers, learning Scratch can be done in conjunction with the teaching of Scratch during computer class.
Adoption	Use Scratch to support traditional instruction. Teachers can create Scratch presentations used to illustrate an instructional idea, or Scratch simulations can be used to demonstrate concepts. Scratch made quizzes can be used for assessment.
Adaptation	Start having students use Scratch more frequently in other subject areas other than computer class. For example simple multimedia presentations, like storytelling in English class.
Appropriation	A focus on cooperative, project-based, and interdisciplinary work using Scratch. Students can collaborate in small groups to design a Scratch project on a given topic, using materials they research and provide. For example, in social studies students can use Scratch in the classroom for projects illustrating their points of view.
Invention	There should now be a shift from teacher-centered instruction to student-centered instruction. Essentially Scratch becomes a tool that students can choose to use for accomplishing tasks, solving problems, and constructing knowledge in all subject areas.

(Apple Computer, Inc., 1995)

Integration Ideas

Scratch projects:

> •can be used by educators to support curricular objectives in academic subjects across the curriculum.
> •can involve the incorporation of more than one subject area of the curriculum.

Subject specific ideas:

Art
Scratch supports the arts by enabling students to create projects that include elements of music, design, drawing, and dance. A virtual museum is a good example of a way to explore Art with Scratch.

A virtual museum is a collection of digital information resources; that is essentially a collection of anything that can be put into digital format.

> They were first used in Education as an alternative to written art history reports; they can also used to further students knowledge of curricular objectives in academic subjects in addition to art (Keeler, n.d.).

With Scratch students have the ability to make virtual museums even more interesting and interactive, while developing their programming skills. See an example of a virtual museum at: http://www.scratchprogramming.org/video.php?vid=28. Note: You can use templates of completed museum projects and have students re-mix them.

Mathematics

Scratch can be used to support the teaching and learning of the elementary maths curriculum covering areas like algebra, numbers, shapes and space, measures and data. Have students create projects which support concepts,

content and skill development; Scratch projects can be used to simulate real world problems. See video of a fraction filler project which is both an example of a Mathematics project that supports skill development and an example of real world problem to which fractions can be applied, at: http://www.scratchprogramming.org/video.php?vid=29

Language Arts

In Language Arts there are many opportunities for improving student writing through the use of Scratch. By creating animations or interactive stories learners can develop their grammar, storytelling and creative writing abilities. Students can also develop their public speaking skills by presenting their animated stories to the entire class.

Students also develop their multimedia skills by drawing characters for their stories, downloading and editing images that they find on the Internet, and importing or recording sounds or music for their stories.

To create animation stories students can first build stories using storyboards(sequence of drawings with dialogue or story) and then convert them into animations. With Scratch "say and think commands", students can easily create written speech bubbles for their story characters. Students can even create interactive stories by using the Scratch "ask command" which prompts users to enter dialogue.

Stories are also a great way to further students knowledge in other curricular areas. See a video example of a story created by a 12-year-old boy in Bangalore who was studying the layers of the Earth in school at:
http://www.scratchprogramming.org/video.php?vid=30

Project Examples

There are many examples of subject specific project galleries available on the Scratch website. Just search the site. Below are but a few examples, note you will leave this site when you click on the links below:

Projects in Science	http://scratch.mit.edu/galleries/view/15003
Math Projects	http://scratch.mit.edu/galleries/view/6423
Best Science (Kids)	http://scratch.mit.edu/galleries/view/36449
Journalism projects	http://scratch.mit.edu/galleries/view/7512
Interactive Reading Project	http://scratch.mit.edu/galleries/view/61659
Book Reports and Projects	http://scratch.mit.edu/galleries/view/9706
Learning Languages	http://scratch.mit.edu/galleries/view/60538

Scratch Project Rubrics

Project rubrics outline the criteria used to evaluate student work. A Scratch project rubric can be generated by the teacher or together with students. Rubrics are easy to use and helpful:

- Tools for both teaching and assessment.
- In allowing students to become more thoughtful when judging the quality of their own Scratch projects and other Scratch projects.
- In reducing the amount of time spent evaluating student work.
- In allowing teachers to evaluate students of all abilities including students. who are gifted and those with learning disabilities.
- In explaining evaluation to students.

(Goodrich, 1997)

Creating a class generated Scratch Rubric

1. Get students to check out projects on the Scratch website and identify what qualities made for a good project. It is easy to find the good and bad projects because statistics are kept for each uploaded project. A popular project will have lots of love-its, re-mixes and downloads. At the Scratch website view featured Scratch projects at: http://scratch.mit.edu/channel/featured

2. Together with students list some of the characteristics of a good project. For example some of the good project qualities: the project works well, easy to use, easy to understand, creative, fun to play, funny, cool, advanced scripts, cool sprites and backgrounds, creative drawings, and creative stories.

3. Use student feedback to create categories for evaluation. For example project design/creativity, user friendliness, programming, backgrounds and sprites.

4. Come up with different levels of quality. For example: excellent, good, average and needs more work.

5. Create the rubric keeping in mind discussions of common problems and the qualities of good and not so good projects.

6. Using the freshly created rubric, chose several projects and evaluate them in groups or as a class.

(Goodrich, 1997)

An example of a class generated Scratch rubric:

	Excellent	**Good**	**Average**	**Needs more work**
Project Design/ Creativity	Project is very creative and clearly demonstrates unique ideas. Well written advanced design.	Creative and has a unique design.	Somewhat creative and unique ideas. Some project design may have been copied from other projects.	Project incomplete.
User friendliness	Project is extremely user-friendly.	Project is user friendly and easy to understand.	Project is not so user-friendly, some parts are not easy to understand.	Project is not user friendly. Difficult to understand what it does or how to use it.
Programming (Scripts)	Scripts are all working, very well designed and using advanced programming techniques. Student has very good understanding of scripts.	All scripts are working and the student understands all the scripts.	Scripts may have some errors and do not work perfectly. Student does not understand some of the scripts.	Scripts do not work.
Backgrounds and Sprites	Are all named properly, and very well designed. They fit together very well making the project look like an advanced design.	Are all named properly and blend nicely together to enhance the project design.	Some have not been named and named and do not blend well into the project.	They have not been named correctly. They are designed poorly and distract from the project design.

References

Alber R.(2011). Six scaffolding strategies to use with your students. Retrieved October 23rd, 2012 from: http://www.edutopia.org/blog/scaffolding-lessons-six-strategies-rebecca-alber

Apple Computer, Inc. (1995). Changing the conversation about teaching, learning, & technology: A report on 10 years of ACOT research. Cupertino, CA: Apple Computers, Inc.

Aronson, E.(2008). Jigsaw classroom. Retrieved October 21, 2012, from http://www.jigsaw.org/.

Chen-Chung L., Yuan-Bang C., & Chia-Wen H.(2011). The effect of simulation games on the learning of computational problem solving, Computer and Education, vol. 57, pp. 1907-1918, 2011.

Crook, S.J. (2009). Embedding scratch in the classroom. Redware Research Limited. Retrieved October 22nd, 2012 from: http://scratch.redware.com/content/embedding-scratch-classroom

Csikszentmihalyi, M., (1996). Creativity: Flow and the Psychology of Discovery and Invention. HarperCollins Publishers, Inc.

Dabbagh, N. (2005). Pedagogical models for E-Learning: A theory-based design framework. International Journal of Technology in Teaching and Learning, 1(1), pp. 25-44.

Goodrich, H. 1997. Understanding rubrics. Educational Leadership 54 (4): 14-17. Retrieved Nov. 4, 2012, from:http://learnweb.harvard.edu/alps/thinking/docs/rubricar.htm

Keeler C.(n.d.). Educational virtual museums developed using powerpoint. Christy Keeler's Homepage. Retrieved November 3rd, 2012, from: http://christykeeler.com/EducationalVirtualMuseums.html

Kordaki M.,(2012). Diverse categories of programming learning activities could be performed within Scratch. Procedia -Social and Behavioral Sciences 46, 1162-66.

Leikin R., Levav-Waynberg A., Gurevich I. & Mednikov L. (2006). Implementation of multiple solution connecting tasks: Do students' attitudes support teachers' reluctance? Focus on learning problems in mathematics, 28(1), 1-22.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E. (2010). The scratch programming language and environment. Transactions on Computer Education 10(4), 1-15

Monroy-Hernandez, A., & Resnick, M. (2008). Empowering kids to create and share programmable media. Interactions, 15(2), 50-53.

National Research Council (1999). Being fluent with information technology. Washington, DC:National Academy Press.

National Research Council (2000). How People Learn: Brain, Mind, Experience, and School. Washington, DC: National Academy Press.

PBS LearningMedia (2012). National PBS survey finds teachers want more access to classroom tech[Press release]. Retrieved on October 11th , 2012 from: http://www.pbs.org/about/news/archive/2012/teacher-survey-fetc/

Resnick, M. (n.d.) Learning by Designing. Retrieved October 21st, 2012, from http://info.scratch.mit.edu/Educators

Rusk,N.,Resnick, M.,& Maloney,J.(n.d.). Learning with scratch:21st century learning skills. Retrieved Oct. 12, 2012 from M.I.T., Lifelong Kindergarten Group Web site: http://info.scratch.mit.edu/sites/infoscratch.media.mit.edu/docs/Scratch-21stCenturySkills.pdf

Theodorou, C. & Kordaki, M. (2010). Super Mario: a collaborative game for the learning of variables in programming. IJAR, 2(4), pp. 111-118.

Woodbridge, J. (2004). Technology integration as a transforming teaching strategy. Technology and Learning. Retrieved Nov. 1st, 2012 from: http://www.techlearning.com/features/0039/technology-integration-as-a-transforming-teaching-strategy/41670