General Information:

- This document was created for use in the "Bridges to Computing" project of Brooklyn College.
- You are invited and encouraged to use this presentation to promote computer science education in the U.S. and around the world.
- For more information about the Bridges Program, please visit our website at: [http://bridges.brooklyn.cuny.edu/](http://bridges.brooklyn.cuny.edu/)

Disclaimers:

- All images in this presentation were created by our Bridges to Computing staff or were found online through open access media sites and are used under the Creative Commons Attribution-Share Alike 3.0 License.
- If you believe an image in this presentation is in fact copyrighted material, never intended for creative commons use, please contact us at [http://bridges.brooklyn.cuny.edu/](http://bridges.brooklyn.cuny.edu/) so that we can remove it from this presentation.
- This document may include links to sites and documents outside of the "Bridges to Computing" domain. The Bridges Program cannot be held responsible for the content of 3rd party sources and sites.
Introduction to Game Programming & Design

Lecture 1: Video Games and Scratch
Game Unit Questions

1. What defines a computer/video game.
2. What do you need to know to make a computer/video game?
3. What make a game fun?
4. What problems are common to video games?
5. What problems are unique to video games?
Content

1. The Complexity of Games
2. Multimedia and Games
3. Programming Languages Review
4. Scratch
The Complexity of Games

- Video games are very complex programs.
- Even a simple FLASH based web game requires a very high level of programming knowledge; may also require expertise in:
  - Imperative, Procedural, OO, Reflective Programming Paradigms
  - Network Protocols
  - Data retrieval and management techniques
  - Graphic & Sound creation/management
  - Geometry, Trigonometry and Calculus
  - Vector mathematics
  - 3D mathematics
Multimedia and Games

- Games are THE quintessential example of multimedia programs:
  - Input to the game can be text, audio, video, 2D and/or 3D constructs as well as analog and digital controller information.
  - Output can be in any of those same but MOST video games are interactive graphics based games.
  - THUS everything we have covered in previous units applies to games as well (especially interface design).
Programming Languages Review

1. What is a language?
2. What are syntax and semantics?
3. What is an IDE (integrated development environment)?
4. What is a paradigm?
5. How does the Imperative Paradigm work?
Programming Languages

- Programming Languages:
  - Allow us to "talk" to a computer, in a language that we can understand - computers only understand binary (0, 1).
  - All languages (to be useful) require a well-defined syntax and semantics:

- Syntax -> Refers to the rules of grammar, word order and punctuation that must be used. A "syntax error" is usually a punctuation error.

- Semantics -> Refers to the meaning of words that are included in a language. Some words have set meanings, others can be changed (variable mouseY).
Programming Languages

- Allow us reuse and share code.
  - We can reuse other programmers functions
  - A library is a collection of functions and variables that can be

- Fall into different paradigms:
  - Paradigm is a structured approach to solving a problem.
  - If you already know one language that uses a particular paradigm it's easier to learn other languages of that same type.
Introduction to SCRATCH

- Scratch is an IDE (Integrated Development Environment) application.
  - An IDE is a program that allows users to write, run and debug other programs.
- Users create programs in Scratch using an imperative, procedural, object-oriented programming language.
- SCRATCH is a visual programming language that uses a very simple syntax.

* Don't worry, all of these terms will be made clear to you.
Let's take a break, and play some games in SCRATCH.

Your instructor will show you how to:

◦ Start Scratch & Load a program
◦ Start and Stop a program
◦ Find specific parts of the Scratch interface (a picture of the interface is on the next slide).

◦ Example Program: http://bridges.brooklyn.cuny.edu/collegenow/modules/P3_GameProgDesign/OtherMaterials/ExampleGames/Ex_SimpleMissileCommand.sb
Scratch Interface

**Scratch Interface**

**Blocks Palette**
- Blocks for programming your sprites.

**Scripts Area**
- Drag blocks in, snap them together into scripts.

**Share**

**Sprite Rotation Style**

**Current Sprite Info**

**Tools**

**Sound**

**Variables**

**Save Language**

**Tabs**
- Edit scripts, costumes, or sounds.

**Toolbar**

**View Mode**
- Change to large or small stage view.

**Presentation Mode**
- Present your project.

**Green Flag**
- A way to start scripts.

**Stop Sign**
- Stops all scripts.

**Stage**
- Where your Scratch creations come to life.

**Mouse X-Y Display**
- Shows location of cursor.

**New Sprite Buttons**
- Create a new character or object for your project.

**Sprite List**
- Thumbnails of all your sprites. Click to select and edit a sprite.
Why use Scratch?

1. Scratch is FREE!
2. Simple development environment.
4. Large library of functions.
5. Can be used to teach basics of programming in 3 important paradigms.
6. Can be used to create ANY simple computer game.
1. Scratch is FREE

- Scratch is developed by the Lifelong Kindergarten group at the MIT Media Lab, with financial support from the National Science Foundation, Microsoft, Intel Foundation, Nokia, and MIT Media Lab research consortia.
- Scratch is free software and will run on Windows, Mac and Linux machines.
- You can download Scratch here:
  - http://scratch.mit.edu/
- The Scratch website has many helpful tutorials as well as a forum for asking questions and getting help.
2. Simple Development

- Scratch requires very little typing in order to create programs.
- Visual Programming (code creation) using drag and drop.

![Scratch Interface](image-url)
3. Simplified Syntax

- No missing semi-colon problems.
- Code blocks can only fit together in pre-defined way.
4. Large library of functions

- Over 100 predefined functions and limited ability to make more.
- Functions cover vast majority of things that you would want a sprite (object) to be able to do in a game.
- Many of these functions simplify complex mathematical problems like object-overlap detection and response.
5. Basics of Programming

• Just like spoken languages programming languages can be categorized into certain types (paradigms).
• 3 of the most popular programming paradigms are:
  A. Imperative -> A 'smart' list.
  B. Procedural -> Making phone calls.
  C. Object Oriented -> Programming with objects.
A. Imperative Programming ( "a smart list" )

- The imperative paradigm is like giving the computer a list, which tells it step-by-step what to do.

- To be "smart" your list needs 3 things:
  i. **Sequence** -> A predefined order in which to process information. (English vs. Hebrew)
  ii. **Selection** -> The ability to make a choice. The "IF" statement.
  iii. **Repetition** -> The ability to repeat an action. The "WHILE" statement.
i. Sequence

- All "scripts" are processed from top down.
- 4 possible start blocks (ways to start a script), and 3 end blocks.
ii. Selection

- Selection means the ability to make a choice
  - Specifically, perform different actions, based on some condition.
- SCRATCH supports the If, If-else and wait_until functions.
iii. Repetition

- Repetition means "looping" doing things over and over (again, possibly only while a certain condition is true).
- SCRATCH supports a variety of looping functions including repeat_until.
B. Procedural Programming ("making phone calls")

- The procedural programming paradigm is based upon the concept of the “procedure call”: the ability to “send a message” to another section of a program.
- Procedural programming allows us to create sections of code that can be reused over and over.
Scratch allows users to send "broadcasts" which can activate other scripts who are listening for a particular broadcast.

```
when I receive "triple value of x"
set [x v] to [x * 3]
stop script
```

This example is trivial. But you could easily have a procedure with a much more complicated function. And coding it this way would mean you only have to write the procedure once. Then call it when you need it.
C. Object-Oriented

• **OO** programming is an extremely important programming paradigm.
• **Scratch** is not true **OO** programming, but good example of basic concepts:
  ◦ Creating programs that are composed of **INTERACTING OBJECTS**.
  ◦ These objects have associated **FACTS** (properties) and **FUNCTIONS** (methods).
Objects in Scratch are called "Sprites".

Properties of Sprites include:
- Location \((x, y)\)
- Look (costume)
- User defined properties (variables).

Functions of Sprites include:
- Move \((go \ to \ x, \ y)\)
- Make Sound (play sound)
- Detect Collision (is touching)
Sprites

- Found in the lower right corner of the IDE screen.
- Can create using templates (with pictures) or draw your own.
- Once a sprite is created you must click on it to select it, THEN you can change its behavior by using the tabs in the main display window.
A SCRATCH game will consist of multiple sprites.
  ◦ NOTE: The STAGE (the display window) is considered to be a special type of sprite.

Each sprite will have its own scripts which will tell it what to do.
  ◦ Example: All sprites will have a "green flag" script which will tell it what to do at the start of the game.

Sprites can "communicate" by making "broadcasts" which can make other scripts run on other sprites.
6. Create ANY simple game.

- At this time Scratch does not support Vector Graphics, Multi-threading, and user library creation.
- Despite these restrictions it is still possible to create some very interesting and exciting applications/games with Scratch.
- ANY simple arcade game or older console game can be recreated in Scratch.
- Many simple browser-based games (Flash) can also be emulated with Scratch.
Example Games

Nuclear HenHouse

Traffic Jam

Space Bounty

Super Mario

Zelda

Donkey Kong
More example games

**Within Scratch checkout:**
- Projects/Games/BugsOnAPlate
- Projects/Games/MarbleRacer

http://scratch.mit.edu/MrMeyer

- Note: With an account you can post your own games and projects on the Scratch website.

http://bridges.brooklyn.cuny.edu/collegenow/scratch.htm

- Here you will find a host of tutorials, as well as templates to help you create games in several common genres.
Are you an Artist?
(not necessarily a 'gamer')

- Using Sketch you can develop, movies, animations and visual storyboards.
  - What's your story?
  - Who are the characters?
  - What's the conflict?
  - What do you want your audience to feel?
- What objects would you need to create?
- What would their properties/functions be?
Artists (cont)

Checkout:
- /Projects/AnimationDayDream
- /Projects/Stories/Stargate
- /Projects/SpeakUpAgainstSpammers
The End