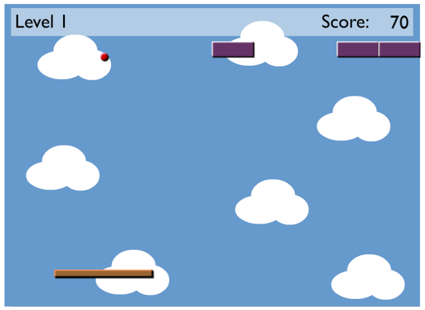
CSCI 1106
Animated Computing



Administrivia

- Instructor: Thomas Trappenberg
- Email: tt@cs.dal.ca
- Meeting Times:
  - Lectures in CS-127: MWF 8:35 – 9:25
  - Labs in CS 134 (Teaching Lab 4):
  - Office hours: I make myself available after the lectures or write email for appointment
  - Course Website: https://projects.cs.dal.ca/hallab/CSCI1106_%282015%29
    All materials including lecture slides are found here.

What is Animated Computing?

an·i·mat·ed (adjective)

full of life or excitement;
lively.

The World is Dynamic and Uncertain

- Observations:
  - Our environment is changing.
  - Events occur at any time.
  - Things break or don't go as planned.
- Yet, we manage to muddle our way through.
- How do computers muddle their way through?

Muddle Through This

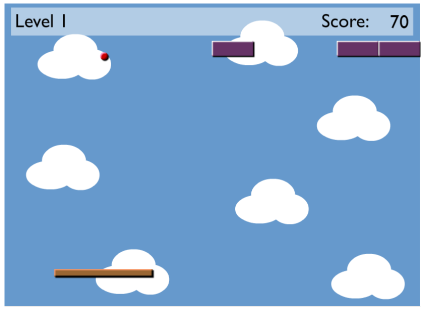
What Animated Computing is about

- A modern world view:
  - Thinking about systems in the real word
  - Realize uncertainty
  - Learn to investigate complex systems

- Some techniques
  - Basic (universal) programing constructs
  - Event driven designs
  - Programing for the real world

Animated Computing with two example projects

- Game Design:
  - Learn about game design
  - Use Scratch to design and develop a game
  - Create a working game of your own design
  - Write a user manual and a technical manual



- Robotics:
  - Learn how robots deal with their environment
  - Use Thymio II robots and the Aseba Studio environment
  - Program a robot to compete in Robot Olympics
  - Write a technical report describing what you did

Robotics vs Game Design?

Games

- Game creates the world
- Composed of sprites
- Position and velocity known
- Collisions are detected
- Goal is to make the world complex

Robots

- World creates the game
- Robot is the sprite
- Position and velocity sensed
- Collisions are avoided
- Goal is to simplify the world

Course Features

- Hands-on introduction to computer science
  - How do computers deal with the real world?
- Two modules: Game Design and Robotics
- In each module you learn a new technology and apply it
- Majority of learning is done in the labs

Labs start tomorrow

- Lectures are mainly to provide context and general discussions

You Learn Best when You

- Learn collaboratively (in small groups)
- Learn by doing (hands-on)
- Learn by figuring things out on your own
- Learn by problem solving

Success at university requires your initiative and self-motivation
Think about why you are here. If you want to learn we are happy to help

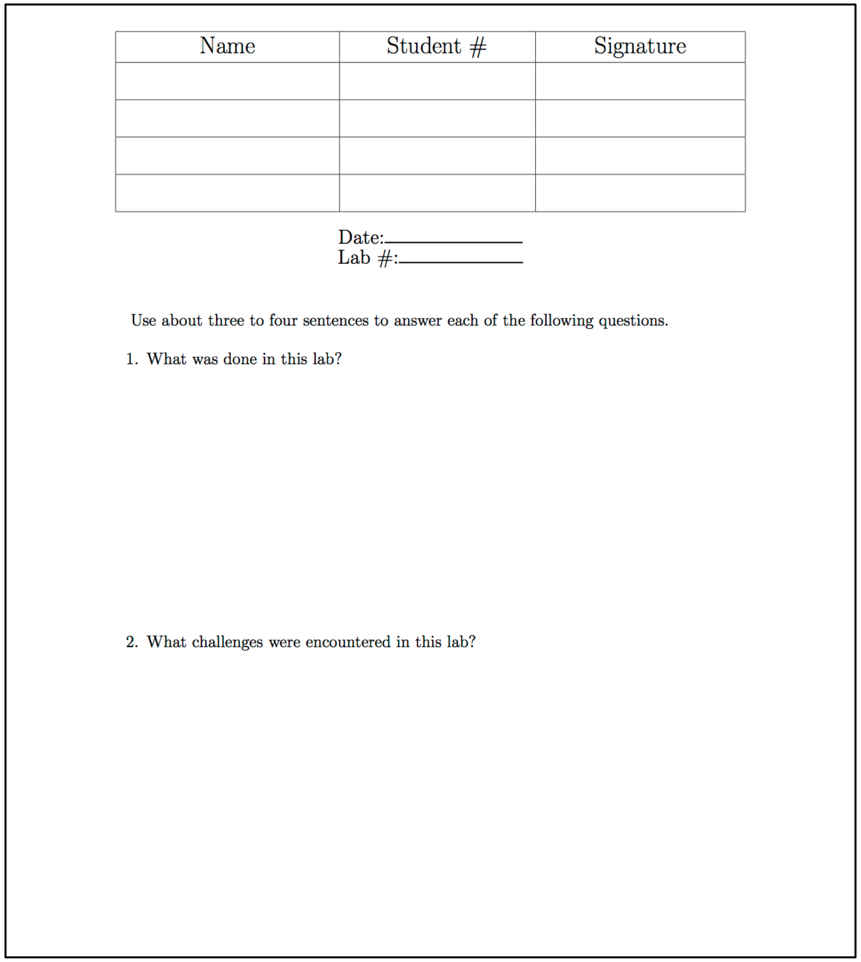
Course Structure: Lectures

- Lectures take place on MWF 8:35 – 9:25
- Monday and Wednesday
  - Cover material you will need for the labs & quizzes
- Friday
  - Every second Friday will be an in-class quiz
  - Other Fridays will be either for guest lectures or catch-up lectures (if needed)
  - You will be informed each Wednesday if there will be a quiz or lecture on the following Friday

Course Structure: Labs

- Labs are where you will do most of the hands-on learning
- You must attend the labs.
  - No changing in lab sections
- In each module (12 labs)
  - First 6 (5) labs are tutorials to learn the technology
  - Next 5 labs are project work periods
  - Last lab is project presentation period
- Each team must submit a lab report at end of the lab
  - Except the presentation period
  - All attending team members are noted on the report
  - **The lab reports require participation from all team members**

Lab Report



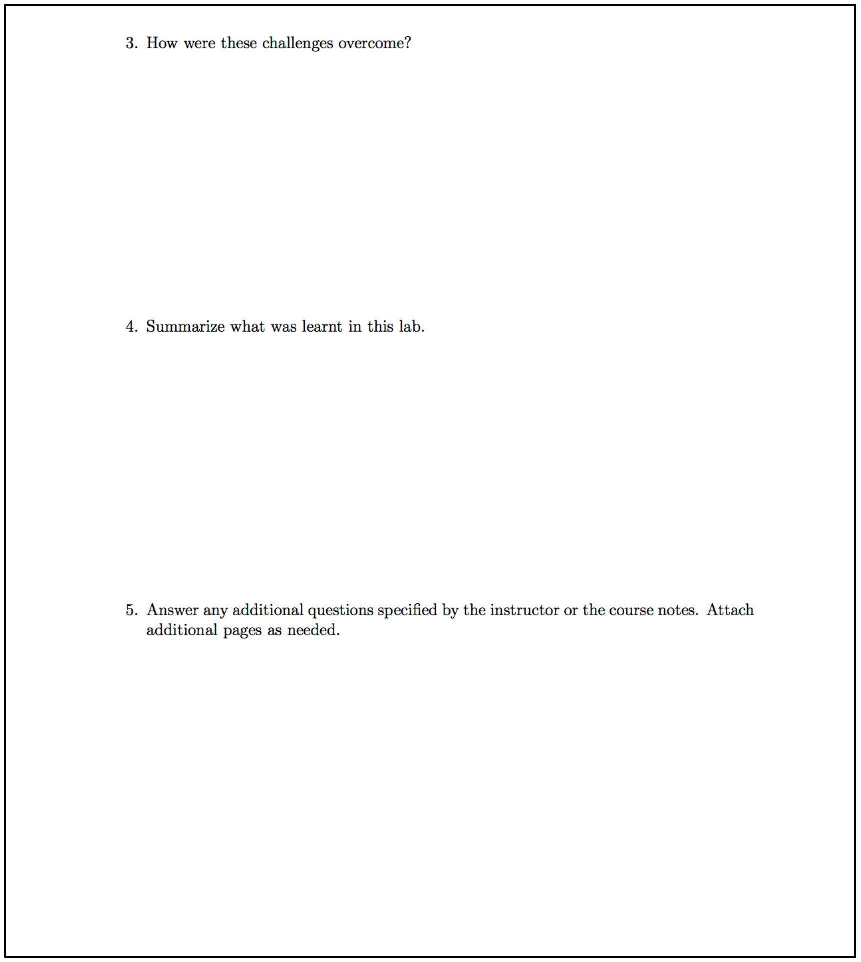
Name	Student #	Signature

Date:____________
Lab #:____________

Use about three to four sentences to answer each of the following questions.

1. What was done in this lab?

2. What challenges were encountered in this lab?



3. How were these challenges overcome?

4. Summarize what was learnt in this lab.

5. Answer any additional questions specified by the instructor or the course notes. Attach additional pages as needed.

Team Work

- Teams are assigned at the start of each module
- Teams comprise usually of four individuals
- Teams are expected to
  - Establish good communication
  - Share equally in the work involved
  - Use individual strengths to benefit the group
- Teams share the same project grade
- Failure to participate in the team may result in an individual's grade being reduced

Evaluation

- Team Work (done in teams of 4 or 5)
  - 20% : Game Design Project
    - 50% : Game Produced
    - 50% : user manual (20%) and technical manual (80%)
  - 20% : Robotics Project
    - 50% : Robot Functionality and Performance
    - 50% : Project Report
  - 10% : Lab Reports (due at end of each lab)
- Individual Work
  - 20% : Bi-weekly Quizzes: Covering labs and lectures
  - 30% : Final Exam

You must pass both the individual and group components to pass the course.

Classroom Rules

- I will start and end classes on time.
- You may come in and leave during class as long as you do not disturb.
- Please let me know if things are unclear. You are here to learn, so do not hesitate to ask.
- You are old enough to decide if you want to attend lectures. Experience shows that there is a large correlation between attendance and success.
- **Please turn cell phone ringers off.**

To Do List

- Make sure your CS account is active.
  - Go to the CS Help Desk on the first floor of the CS building to activate it.
- Log web page and locate the course material and resources.
  - https://projects.cs.dal.ca/hallab/CSCI1106_%282015%29
  - Look over the first tutorial for tomorrow.

Academic Honesty

- Academic integrity means being honest in the fulfillment of your academic responsibilities thus establishing mutual trust.
- Violations of intellectual honesty are offensive to the entire academic community, not just to the individual faculty member and students in whose class an offense occurs.
  - E.g., cheating on tests, plagiarism, falsification of experimental data, etc.
- All cases of academic misconduct are automatically referred to the Faculty Academic Integrity Officer (Associate Dean).

Lets start right away

- First Module: Program a Game
  - We use the program called Scratch that is developed by the MIT media lab.
  - The program is meant to be easy to learn and to use, and the purpose is to encourage creativity.
  - While Scratch is easy to learn thanks to the visual programing environment, it still contains the principles of a modern events-based programming style.

The Game Design Module

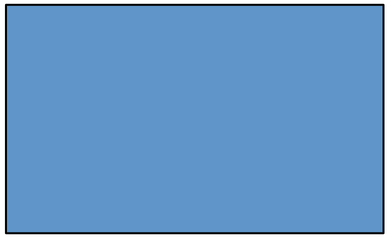
Topics

- Structure of a game
- Game mechanics
  - Collision Detection
  - Player movement
  - Autonomous Game Elements
  - Randomness
  - Controls
- Playability and play testing

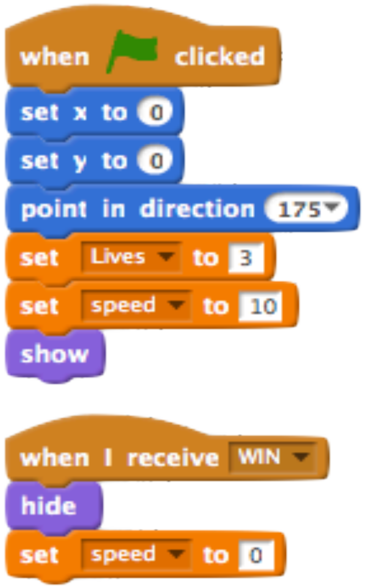
To Do List

- Five tutorials:
  - Implement a game
  - Learn about game design
- One play-testing session
- Game Design Project
  - Design your own game
  - Implement the game
  - Write a technical manual
  - Write a user manual

Scratch in a Nutshell



- A Scratch program consists of
  - A *stage* on which sprites are displayed
  - One or more *sprites*
    - *graphical objects that interact on the stage*
  - Zero or more *scripts associated with the sprites*
- A *sprite* has
  - *Properties such as position, direction, size, etc.*
  - *Zero or more variables* used to store values
  - *One or more costumes*, describing how it looks
  - *Zero or more sounds* that it can emit
  - Zero or more *scripts* that respond to events
- A *script* responds to an event
  - These scripts are also called event handlers



```
when [flag] clicked
set x to 0
set y to 0
point in direction 175
set Lives to 3
set speed to 10
show

when I receive WIN
hide
set speed to 0
```

Next Weeks Tutorial

- Get Familiar with Scratch
- Start programing a Brick-Breaker game