Comp-361 : Game Programming
Lecture 5

Alexandre Denault
Original notes by
Jörg Kienzle and Hans Vangheluwe
Computer Science
McGill University
Winter 2008

Time Slot

Monday	Wednesday
9h45	9h45
10h00	10h00
10h15	10h15
10h30	10h30
10h45	10h45
11h00	11h00
11h15	11h15

The Teams

Team Blown Away	2
Jmonkeys	7
The Admiralty	13
ZombiePirateNinjaMonkey	13
The Gamesters	15
BackShot	36
The Hacks	37
Team Magadath	41
Sons of Liberty	49
Vote*	63
The Broken Rubbers	50
Purple Dinosaurs!	77

Meetings

- McConnell 322
- Be on time!

What you should be doing?

- You should have a team
- You have nominated a team leader
- You should have an initial plan
  - What rules do you want to changes?
  - What extra features do you want to do?
  - How to split the work?
- You should be exploring technologies
- You do not need to have started coding

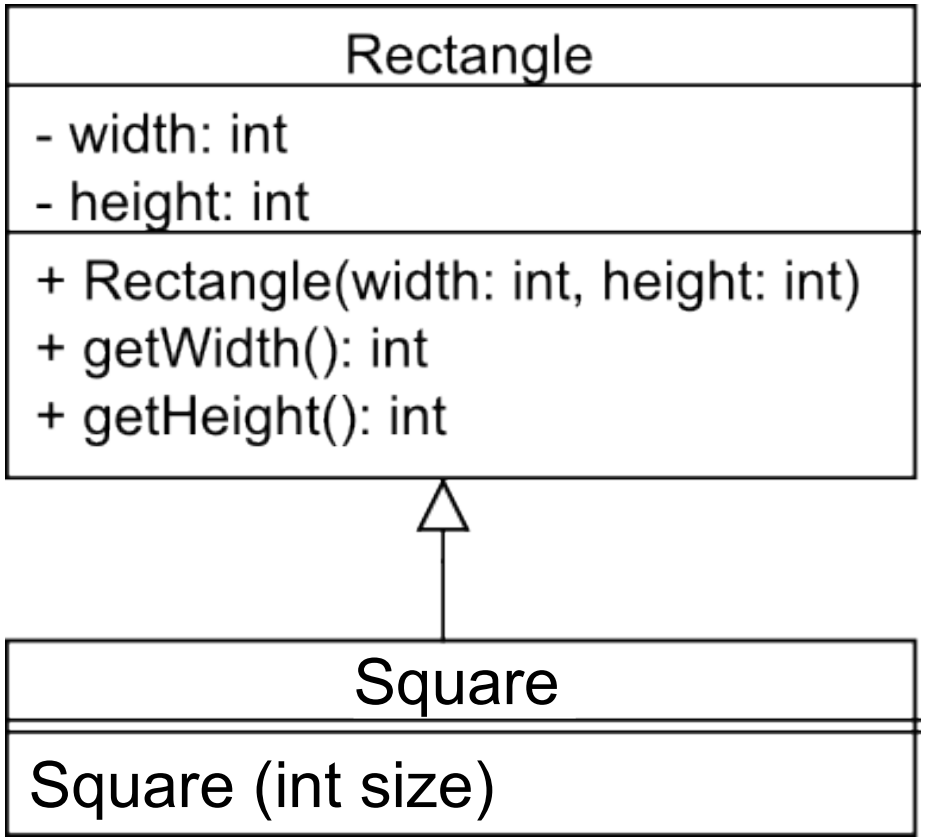
Comp 361 - First Deliverable

- Short (3 or 4 pages) document with the following
  - The name of the team members.
  - How work will be tentatively broken down (with initial timetable).
  - What technology do you plan to use.
  - Any changes you plan to make to the game or the rules.
  - A simple UML diagram describing the main data structures of the game.
  - Two drawings illustrating what Phase 1 and Phase 3 of the game might look like.
- The grading will be based on completeness. This includes
  - All the basic game components should be in the class diagram.
  - The illustration should allow me to understand how to execute the basic actions of the game.
- Due January the 30th, in class
- Late policy:
  - Max grade is reduced by 20% per day late.
  - Hand in only by WebCT only if late.

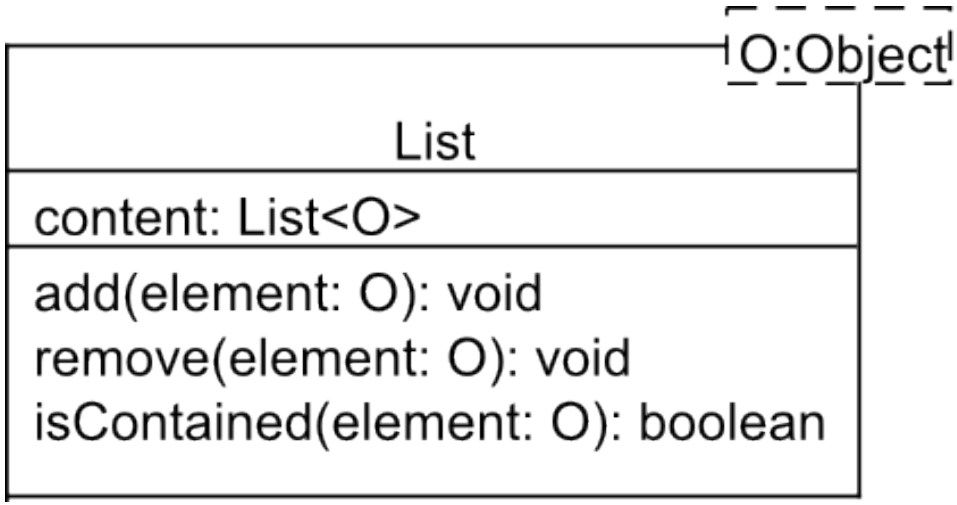
Visibility

- Prefixes for attributes and methods
  - \+ public – visible to any class
  - # protected – visible to any subclass *
  - – private – visible only to class itself
  - ~ package – visible to any class within enclosing package
- Visibility is a class feature. It is found only in class diagrams.

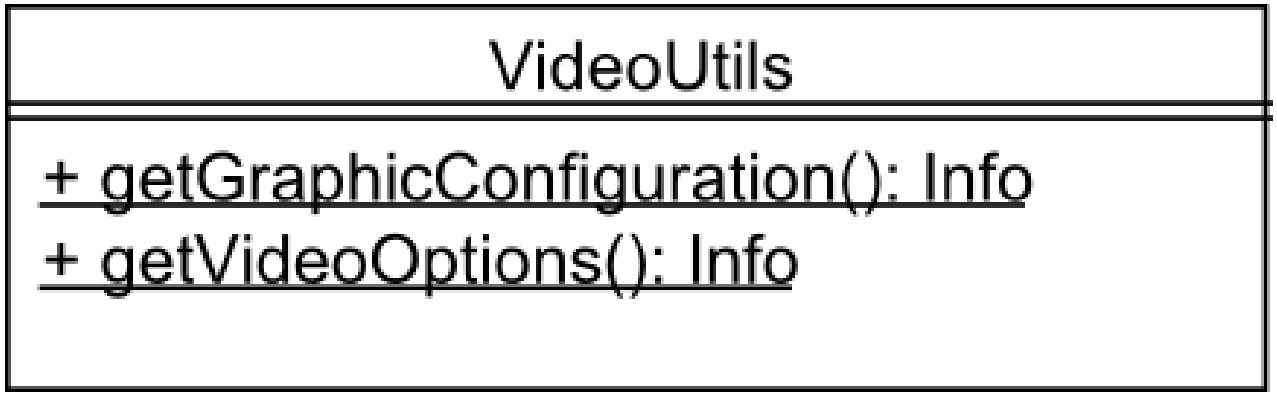
Inheritance



Templates


O:Object
List
content: List<O>
add(element: O): void
remove(element: O): void
isContained(element: O): boolean

Static Members



Minueto In a Nutshell

- Multi platform 2D Graphic Programming Framework for Game Development in Java
  - Windowed and Full-Screen Graphics
  - Loading and displaying images (jpg, png, …)
  - Drawing standard shapes (lines, squares, circles, …)
  - Displaying text
  - Scaling and rotating images
  - Game Input: Mouse and keyboard

Fire in the Sky

Minueto

- Very simple to learn and use
  - “Can be learn in less than an hour”
- Allow a programmer to create a window and draw an image in less than 10 lines of code
- Multi-platform
  - Provided by Java
- Yet provide good performance!

Isn’t Java Too Slow for Games?

- Several games have been successfully ported to Java
  - Quake 2
- Some commercial games written in Java have been released
  - Law and Order, Dead on the Money
  - Bang Howdy
  - Rune Scape
  - Puzzle Pirates

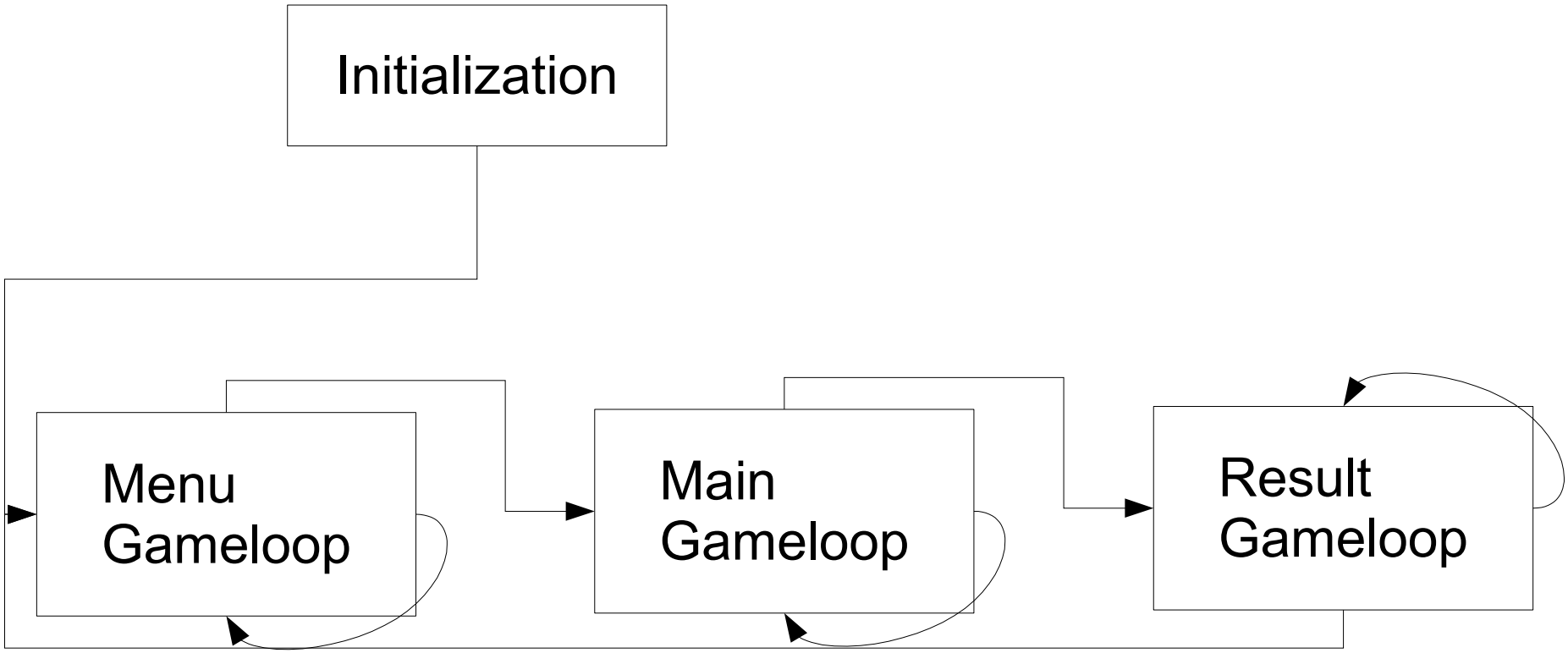
Pong 36-Hours Challenge

- Build a Pong game using Minueto.
  - http://en.wikipedia.org/wiki/Pong
- The game archive (zip) should not be more than 2 Mb.
- Game will be evaluated on look and playability.
- Best game gets a Veto coupon.
- If more than 10 people participate, a Veto coupon will also be drawn.
- Hand-in by WebCT.

Components of Game

- Initialization
  - Create Game Window
  - Load any resources needed by the game
- Gameloop
  - Check for input
  - Update logic of game
  - Render the screen
  - Loop back to start of game loop

Typical Game



Init – Create Window

```
MinuetoWindow mwiWindow = new
          MinuetoFullscreen(640, 480, 32);
```

or

```
MinuetoWindow mwiWindow = new
          MinuetoFrame(640, 480, true);
```

then

```
mwiWindow.setVisible(true);
```

FullScreen

- Standard resolutions
  - 640x480
  - 800x600
  - 1024x768
  - 1280x1024
- Color depth
  - Recommended: 32

Init – Load Resources

```
MinuetoImageFile mimDemoImage;

try {
   mimDemoImage = new
      MinuetoImageFile("strawberry.jpg");
} catch (MinuetoFileException e) {
   System.out.println("Could not load file");
   return;
}
```

Example GameLoop

```
while (true) {
    // handle all input from player
    // and update state
    while (meqQueue.hasNext()) {
          meqQueue.handle();
    }

    // draw the new frame
    mwiWindow.draw(…);
    mwiWindow.render();

    Thread.yield();
}
```

GameLoop – Input

- Input is external stimuli to your game.
- Input can come from:
  - Keyboard
  - Mouse
  - Disk I/O
  - Network

Keyboard Listener

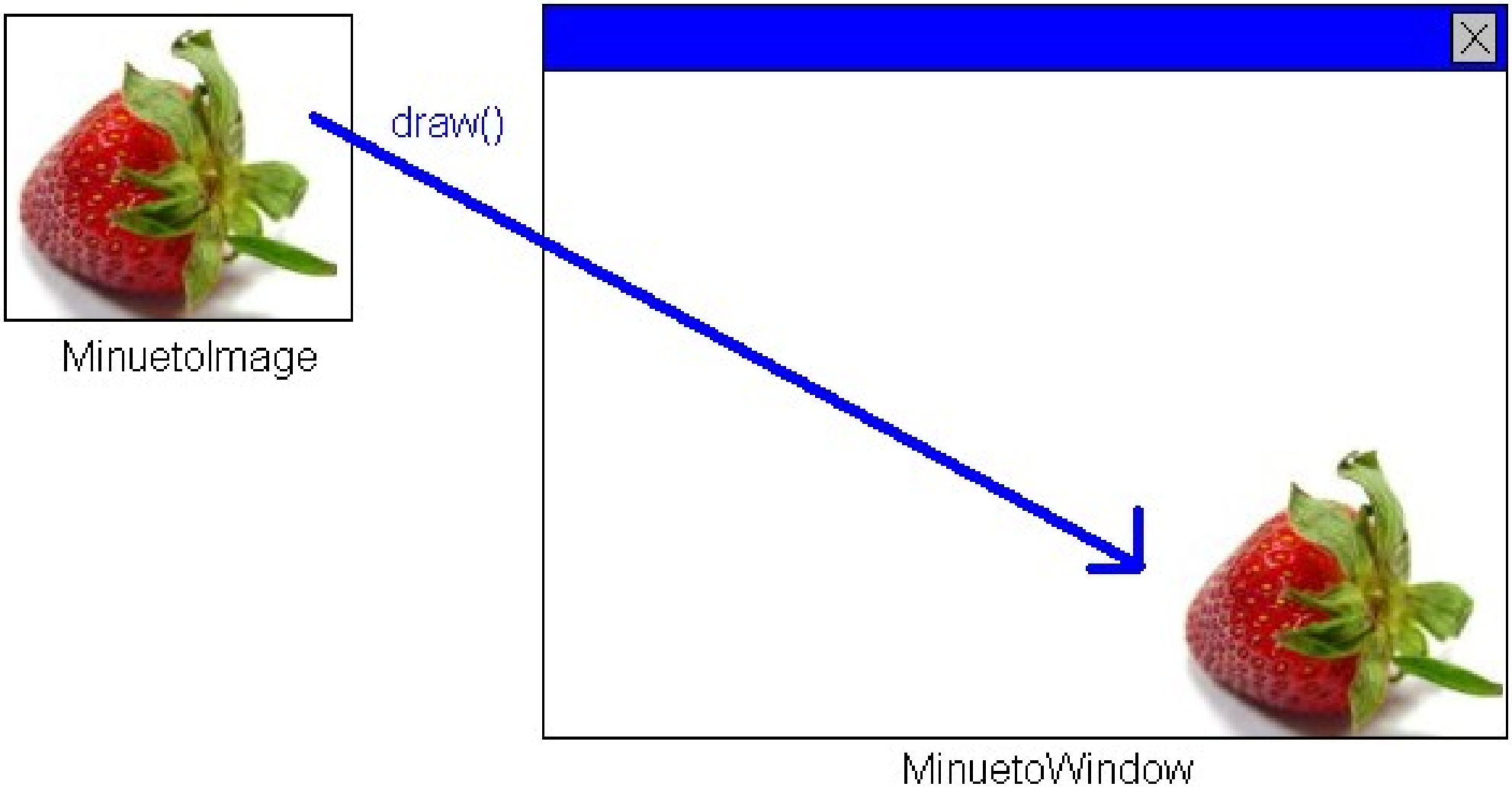
```
public class DemoKeyboardHandler implements
     MinuetoKeyboardHandler {

   public void handleKeyPress(int iValue) { }
   public void handleKeyRelease(int iValue) { }
   public void handleKeyType(
                  int iValue, char keyChar) { }
}

mwiWindow = new MinuetoFrame(640, 480, true);
meqQueue = new MinuetoEventQueue();

mwiWindow.registerKeyboardHandler
(new DemoKeyboardHandler(), meqQueue);
```

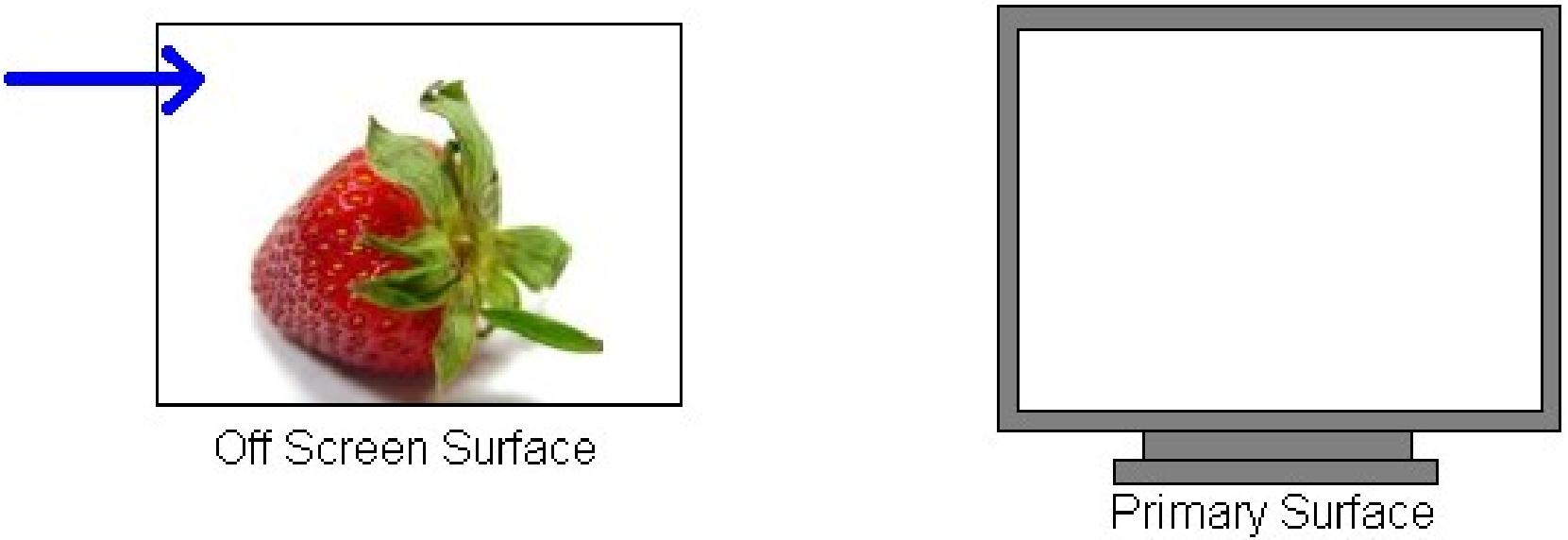
Drawing



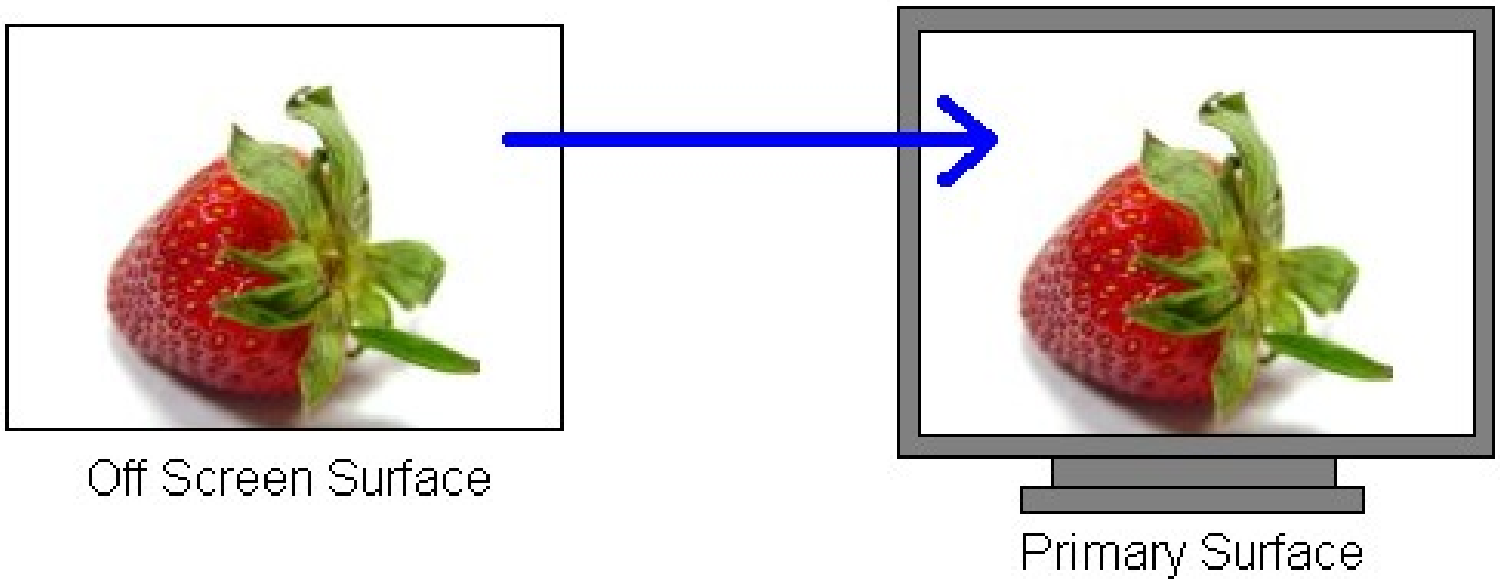

```
mwiWindow.draw(mimDemoImage, 100, 500);
```

Double Buffer

1. Draw on the off screen surface.



2. Copy the content of the off screen surface on the primary surface (screen).



Example GameLoop

```
while (true) {
    // handle all input from player
    // and update state
    while (meqQueue.hasNext()) {
          meqQueue.handle();
    }

    // draw the new frame
    mwiWindow.draw(…);
    mwiWindow.render();

    Thread.yield();
}
```

More Minueto

- Visit http://minueto.cs.mcgill.ca/
  - For step by step instructions, check out the howtos
  - When working with Minueto, the APIs are your best friend
- Download Minueto
  - You'll find 25 samples showing how to use Minueto
  - You also get your own local copy of the API
- Ask the T.A.s
  - It's their job is to help you