EECS 1710
Programming for Digital Media

P10 :: Moving Images

Moving/Changing Images

- Recall → utilize draw loop
  - animating
    - modify a graphic (clear and redraw)
    - modify an image (all or some pixels)

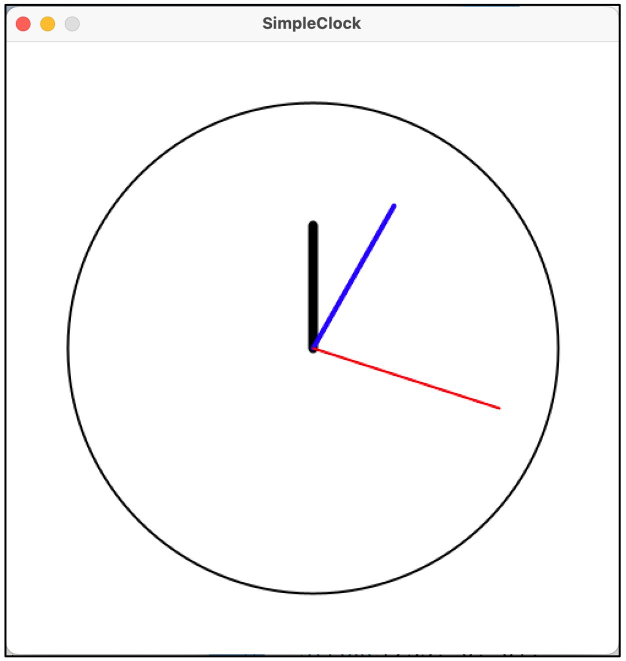
- Capturing & saving frames (as `Pimage[]` or `ArrayList<>`)
  - exporting frames as Gif files

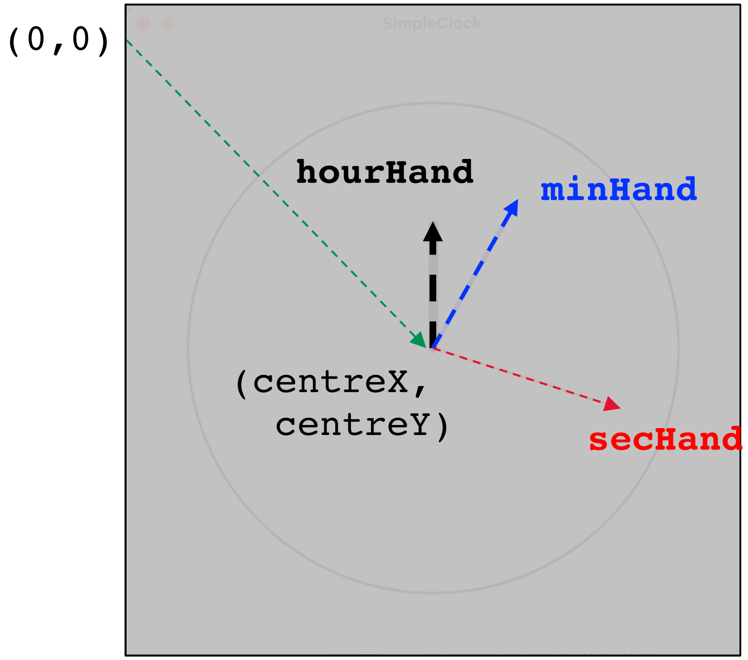
- *Working with video*

Simple clock (animation)

- Using PVectors

PVector hourHand;
PVector minHand;
PVector secHand;





```
// simple clock animation

final float CLOCK_RADIUS = 200;
final float HOUR_RADIUS = CLOCK_RADIUS/2;
final float MIN_RADIUS =  2*CLOCK_RADIUS/3;
final float SEC_RADIUS =  4*CLOCK_RADIUS/5;

PVector hourHand;
PVector minHand;
PVector secHand;

int hr;
int mn;
int sc;
```

```
void setup() {

  size(500, 500);

  hourHand = new PVector(0, -1);        // vectors all pointing up
  hourHand.mult(HOUR_RADIUS);           // scaled to diff sizes

  minHand = new PVector(0, -1);
  minHand.mult(MIN_RADIUS);

  secHand = new PVector(0, -1);
  secHand.mult(SEC_RADIUS);

  frameRate(1);           // framerate = 1 frame per second
                          // (so draw loop can update once per sec)

  hr = hour();            // methods to get current time
  mn = minute();          // from computer's internal clock
  sc = second();

  println("hr:mn:sc = " + hr + ":" + mn + ":" + sc);
}
```

```
void draw() {

  background(255, 255, 255);

  // draw clock boundary (circle)
  stroke(0, 0, 0);
  strokeWeight(2);
  circle(width/2, height/2, CLOCK_RADIUS*2);

  // get current time instant
  hr = hour();
  mn = minute();
  sc = second();

  println("hr:mn:sc = " + hr + ":" + mn + ":" + sc);




  // …
// continued …

  // set vector rotations.  Each position initially at -90 (i.e. up), and
  // adding an angle to move clockwise from there as a percentage of max
  // value expected for that hand.  I.e. hr/12 has hr=12 as max (or hr=23)
  // and will determine a percentage of 360 degrees… note: 23 gives > 360

  PVector.fromAngle(radians(-90 + float(hr)/12*360), hourHand);
  hourHand.mult(HOUR_RADIUS);  // have to rescale as fromAngle makes unit vector
                               // i.e. unit vector = vector of length = 1
  stroke(0, 0, 0);
  strokeWeight(8);
  // each hand is a line from centre to endpoint of each hand's PVector
  // note centreX = width/2; centreY = height/2
  line(width/2, height/2, width/2+hourHand.x, height/2+hourHand.y);


  PVector.fromAngle(radians(-90 + float(mn)/60*360), minHand);
  minHand.mult(MIN_RADIUS);
  stroke(0, 0, 255);
  strokeWeight(4);
  line(width/2, height/2, width/2+minHand.x, height/2+minHand.y);


  PVector.fromAngle(radians(-90 + float(sc)/60*360), secHand);
  secHand.mult(SEC_RADIUS);
  stroke(255, 0, 0);
  strokeWeight(2);
  line(width/2, height/2, width/2+secHand.x, height/2+secHand.y);
}
```

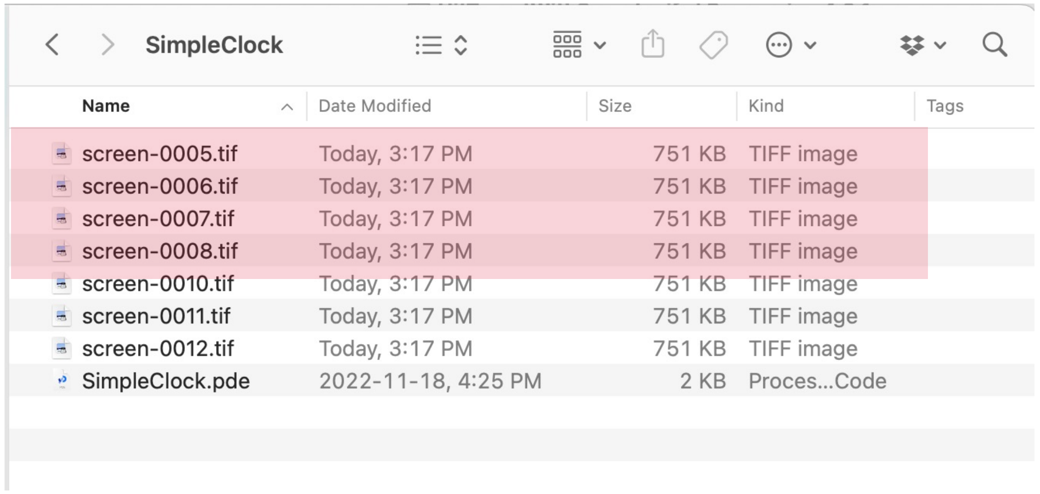
Capturing PImage’s (as frames)

- Save as an *array* (or *ArrayList*) of PImage objects
  - For use later? Or save (directly or later) as images in your sketch folder
    - Images for different frames may be used to reconstruct the animation (as a sequence) in some 3rd party software

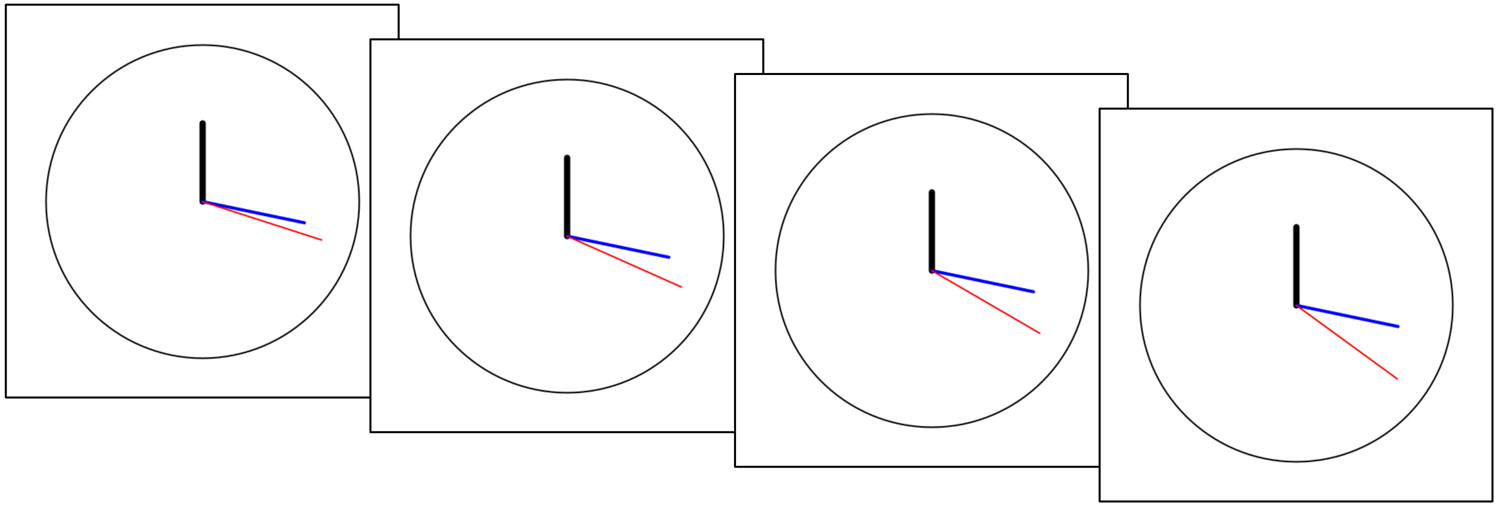
    OR

    - Use a library to create animated files (like Gif animation’s)

```
// global variables (before setup)
PImage[] animation;
int animIdx = 0;


// inside setup (e.g. make a buffer (array) to save 10 frames)
animation = new PImage[10];



// lets assume we save frames on pressing the 's' key on keyboard
void keyPressed() {

  if (key=='s') {

    // only try to save current frame if we haven't filled the animation array
    if (animIdx<animation.length) {

      animation[animIdx] = createImage(width, height, RGB);     // create new PImage
      animation[animIdx].set(0, 0, get());        // get entire image from app window

      saveFrame();              // this will save image from app window
                                // as a file in your sketch folder

      animIdx++;

    }
  }


}
```



Name	Date Modified	Size	Kind	Tags
screen-0005.tif	Today, 3:17 PM	751 KB	TIFF image	
screen-0006.tif	Today, 3:17 PM	751 KB	TIFF image	
screen-0007.tif	Today, 3:17 PM	751 KB	TIFF image	
screen-0008.tif	Today, 3:17 PM	751 KB	TIFF image	
screen-0010.tif	Today, 3:17 PM	751 KB	TIFF image	
screen-0011.tif	Today, 3:17 PM	751 KB	TIFF image	
screen-0012.tif	Today, 3:17 PM	751 KB	TIFF image	
SimpleClock.pde	2022-11-18, 4:25 PM	2 KB	Proces...Code	



...

Name

saveFrame()

Description

Saves a numbered sequence of images, one image each time the function is run. To save an image that is identical to the display window, run the function at the end of `draw()` or within mouse and key events such as `mousePressed()` and `keyPressed()`. Use the Movie Maker program in the Tools menu to combine these images to a movie.

If `saveFrame()` is used without parameters, it will save files as screen-0000.tif, screen-0001.tif, and so on. You can specify the name of the sequence with the `filename` parameter, including hash marks (####), which will be replaced by the current `frameCount` value. (The number of hash marks is used to determine how many digits to include in the file names.) Append a file extension, to indicate the file format to be used: either TIFF (.tif), TARGA (.tga), JPEG (.jpg), or PNG (.png). Image files are saved to the sketch's folder, which may be opened by selecting "Show Sketch Folder" from the "Sketch" menu.

Syntax

```
saveFrame()
saveFrame(filename)
```

Parameters

`filename` `(String)` any sequence of letters or numbers that ends with either ".tif", ".tga", ".jpg", or ".png"

Return

`void`

Related

`save()`

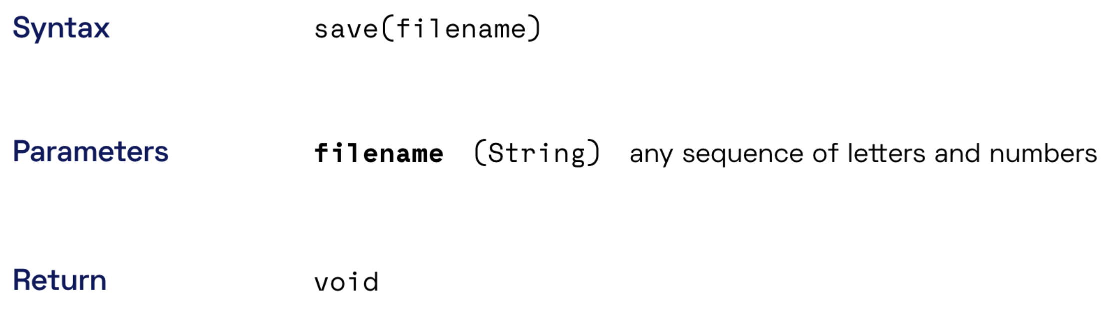
`createGraphics()`

`frameCount`

https://processing.org/reference/saveFrame_.html

Alternatively..

- PImage objects have access to a save() method which does the same thing (though you must specify a file name)
- This can also be called directly (which will save from the app window)
  - all images saved from app window are treated as opaque…
  - if you want to save with alpha, you need to create PImage with createImage() and invoke save() on that object



Syntax	`save(filename)`
Parameters	**`filename`** `(String)` any sequence of letters and numbers
Return	`void`

Saving frames in an ArrayList?

elements stored in these array lists are implied

- `StringList` → uses String objects as elements
- `IntList/FloatList` → uses ints or floats as elements

how about generic ArrayList?

- `ArrayList<PImage>`
    - ArrayList is the version we use with a generic type (we have to specify what the **type** is at declaration and initialization)
    - Here, we want to use the type: PImage

```
ArrayList<PImage> myAList = new ArrayList<PImage>();
myAList.add( // PImage object  );
```

So… essentially :

```
StringList myStrList = new StringList();
```

is equivalent to

```
ArrayList<String> myStrList = new ArrayList<String>();
```

```
IntList myIntList = new IntList();
```

is equivalent to

```
ArrayList<Integer> myIntList = new ArrayList<Integer>();
```

```
FloatList myFList = new FloatList();
```

is equivalent to

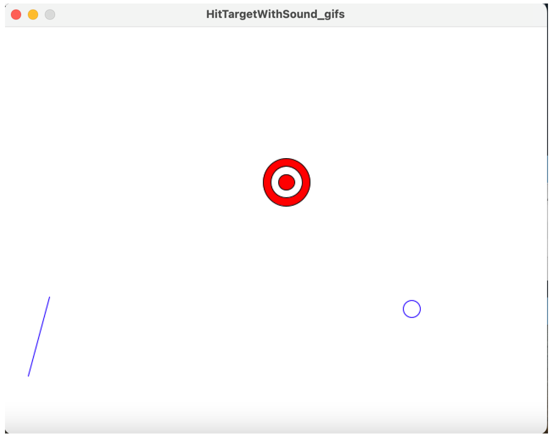
```
ArrayList<Float> myFList = new ArrayList<Float>();
```

In **Processing**, StringList, IntList and FloatList are given to simplify things
In pure **Java**, we only have the generic ArrayList type.

ArrayLists **cannot** store primitive types, only reference types
so there are reference versions of the primitive types..
(i.e. Integer, Character, Float, Double, etc.)

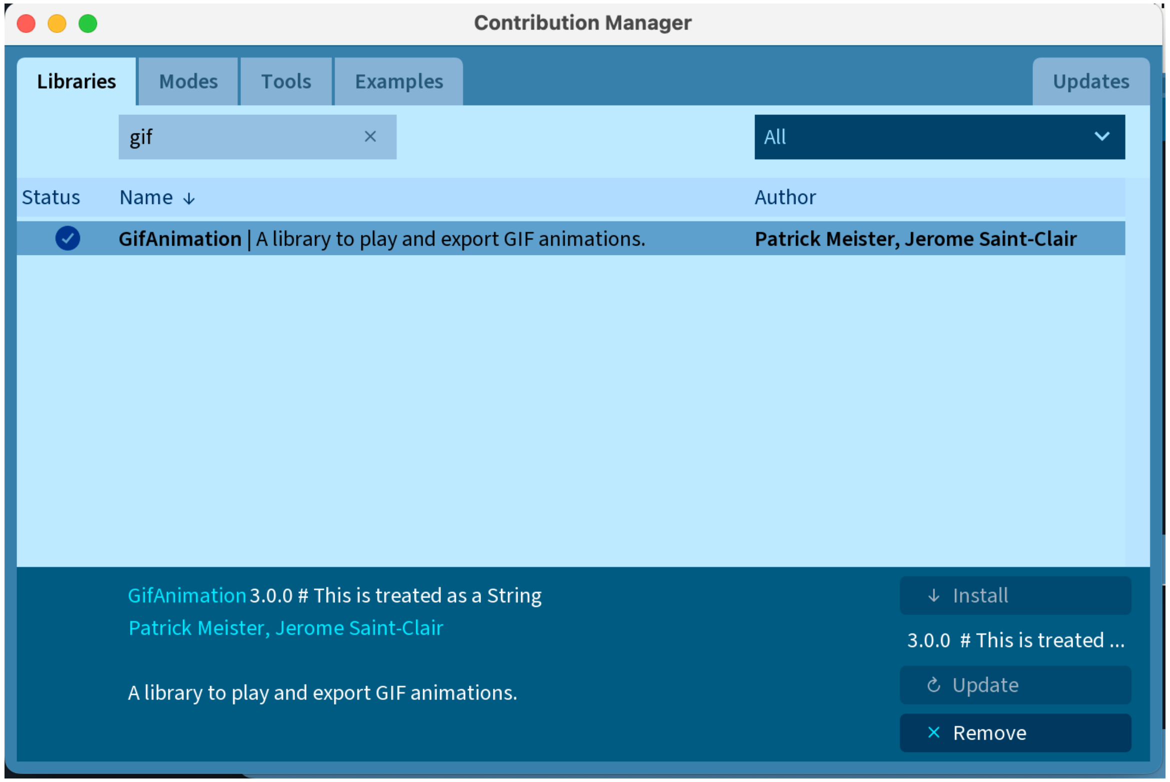
Recall: launching the projectile

- Used draw() method to update projectile position & redraw.
  - We can save regular frames (every 60 frames, or once per second if frame rate = 60fps)



```
ArrayList<PImage> animation;

// inside setup()
animation = new ArrayList<PImage>();

// inside draw()
if (frameCount%60 == 0) {
        PImage thisFrame = get();
        animation.add(thisFrame);

        saveFrame();
          // or thisFrame.save(”frame” + frameCount “.tif“);

}
```

Using the Gif Animation Library (external lib)



gif

Status	Name	Author
✓	**GifAnimation** | A library to play and export GIF animations.	**Patrick Meister, Jerome Saint-Clair**

GifAnimation 3.0.0 # This is treated as a String
Patrick Meister, Jerome Saint-Clair

A library to play and export GIF animations.

3.0.0 # This is treated ...

Using the Gif Animation Library (external lib)

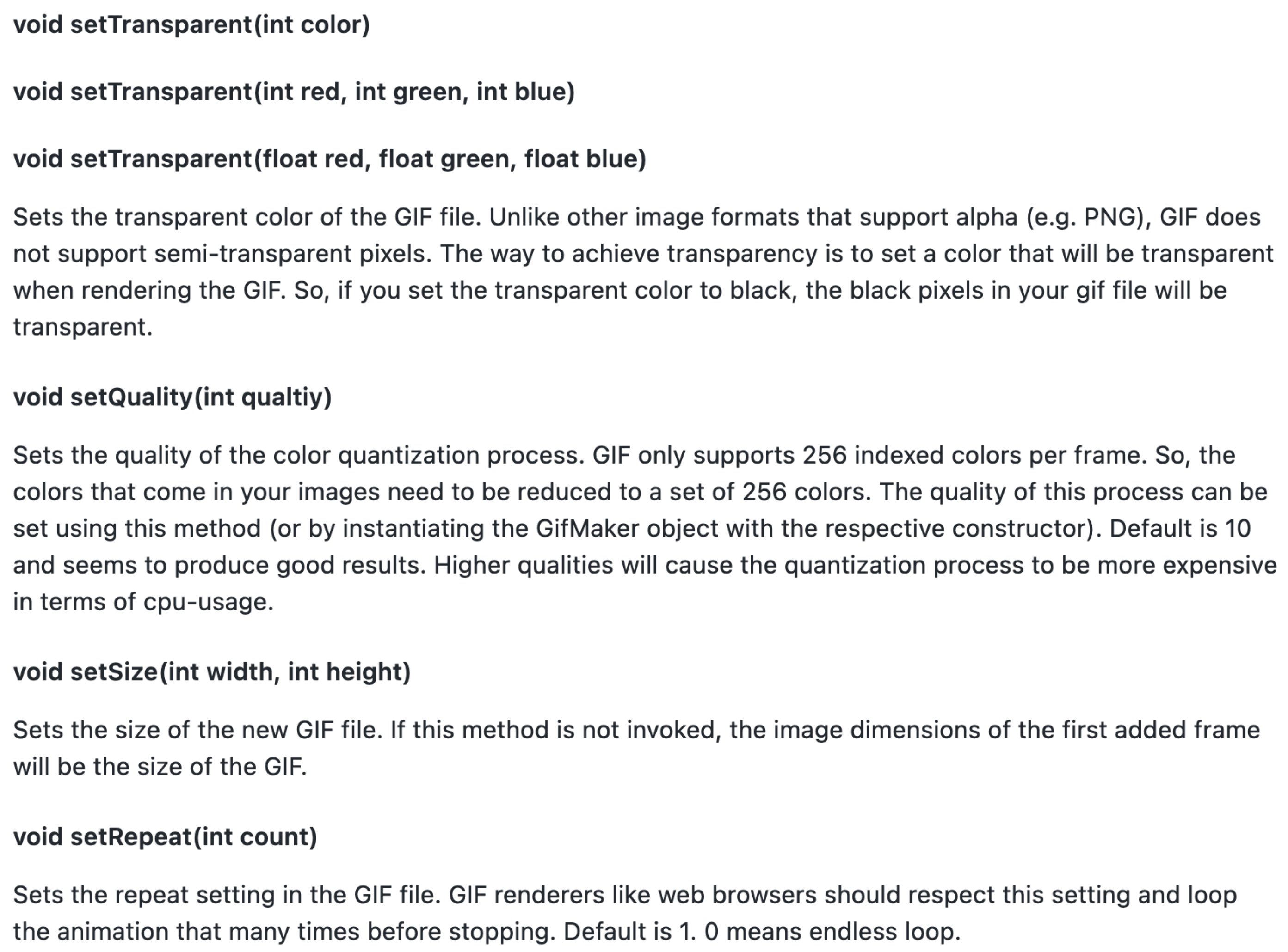
```
import gifAnimation.*;

// TWO MAIN OBJECTS

Gif myAnimation;            // use this for loading in and running
                             // an existing gif file


GifMaker gifExport;         // use this to setup and create a gif
                             // from your app window graphics
```

- Documentation/library hosted on github (as are many projects):
  - https://github.com/extrapixel/gif-animation

Gif Animation (documentation)

- Documentation:
  - https://github.com/extrapixel/gif-animation

GifMaker class:



void setTransparent(int color)

void setTransparent(int red, int green, int blue)

void setTransparent(float red, float green, float blue)

Sets the transparent color of the GIF file. Unlike other image formats that support alpha (e.g. PNG), GIF does not support semi-transparent pixels. The way to achieve transparency is to set a color that will be transparent when rendering the GIF. So, if you set the transparent color to black, the black pixels in your gif file will be transparent.

void setQuality(int qualtiy)

Sets the quality of the color quantization process. GIF only supports 256 indexed colors per frame. So, the colors that come in your images need to be reduced to a set of 256 colors. The quality of this process can be set using this method (or by instantiating the GifMaker object with the respective constructor). Default is 10 and seems to produce good results. Higher qualities will cause the quantization process to be more expensive in terms of cpu-usage.

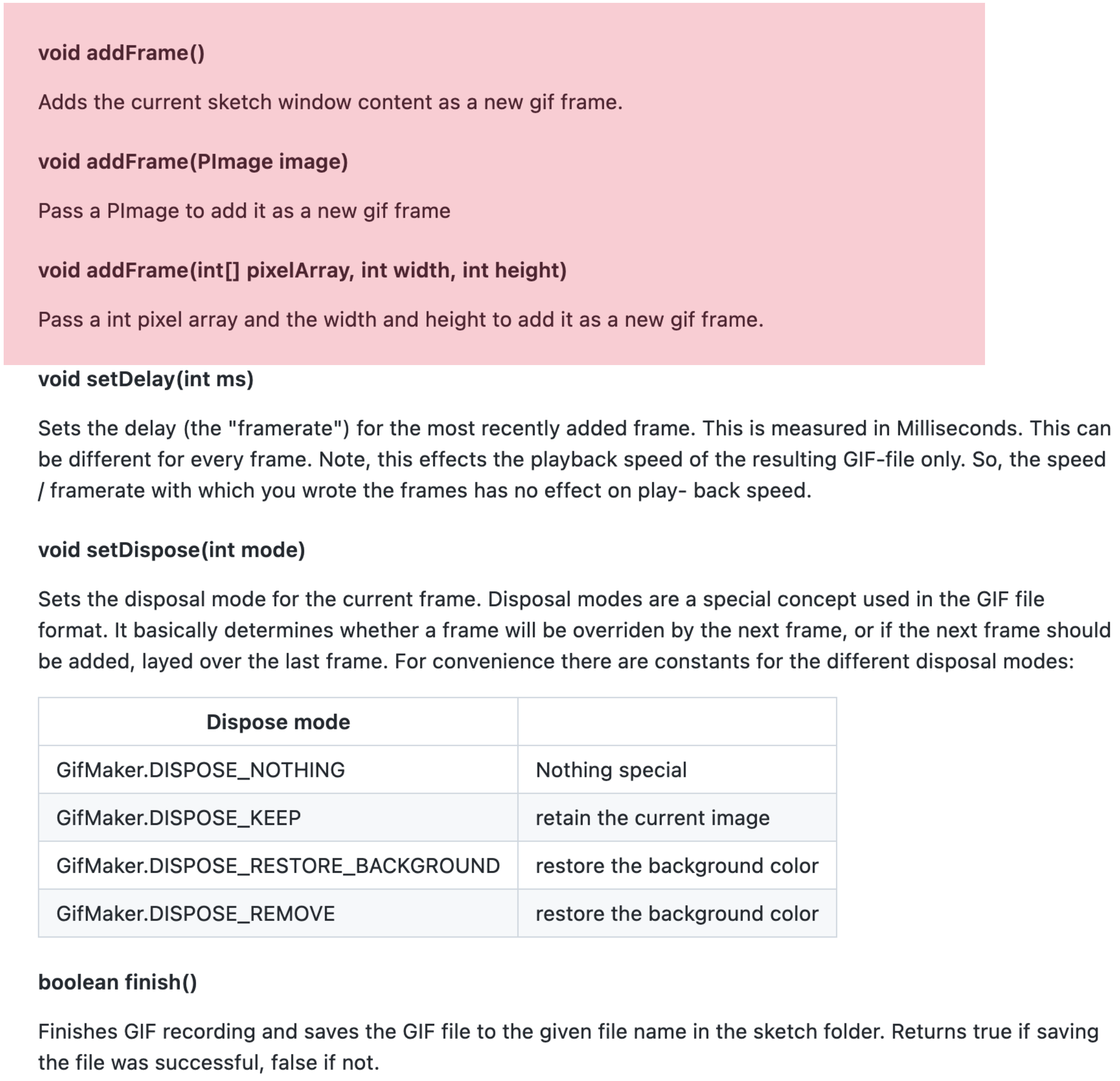
void setSize(int width, int height)

Sets the size of the new GIF file. If this method is not invoked, the image dimensions of the first added frame will be the size of the GIF.

void setRepeat(int count)

Sets the repeat setting in the GIF file. GIF renderers like web browsers should respect this setting and loop the animation that many times before stopping. Default is 1. 0 means endless loop.

3 ways to add frames



void addFrame()

Adds the current sketch window content as a new gif frame.

void addFrame(PImage image)

Pass a PImage to add it as a new gif frame

void addFrame(int[] pixelArray, int width, int height)

Pass a int pixel array and the width and height to add it as a new gif frame.

void setDelay(int ms)

Sets the delay (the "framerate") for the most recently added frame. This is measured in Milliseconds. This can be different for every frame. Note, this effects the playback speed of the resulting GIF-file only. So, the speed / framerate with which you wrote the frames has no effect on play- back speed.

void setDispose(int mode)

Sets the disposal mode for the current frame. Disposal modes are a special concept used in the GIF file format. It basically determines whether a frame will be overriden by the next frame, or if the next frame should be added, layed over the last frame. For convenience there are constants for the different disposal modes:

Dispose mode	
GifMaker.DISPOSE_NOTHING	Nothing special
GifMaker.DISPOSE_KEEP	retain the current image
GifMaker.DISPOSE_RESTORE_BACKGROUND	restore the background color
GifMaker.DISPOSE_REMOVE	restore the background color

boolean finish()

Finishes GIF recording and saves the GIF file to the given file name in the sketch folder. Returns true if saving the file was successful, false if not.

```
// MAKING A GIF FROM FRAMES (instead of storing/saving)


// INSIDE setup() or method where you want to create the gif

gifExport = new GifMaker(this, "export.gif");

gifExport.setRepeat(0);       // parameter = count
                             // (num times to loop before stopping;
                             // 0 = endless)


gifExport.setTransparent(0,0,0);  // sets which colour to set
                                  // to transparent.. e.g. black



// INSIDE draw() or method where you want to build the gif
gifExport.setDelay(1);         // sets time between frames
gifExport.addFrame();          // record app window as a frame
```

YORK UNIVERSITÉ UNIVERSITY

Hit Target example: saving stored frames (record up until hit target then make gif)

```
// inside draw()
if (frameCount%6 == 0) {             // at 60fps, save every 6th (i.e. 10fps)
    PImage thisFrame = get();

    if (animation.size()<100)
      animation.add(thisFrame);      // add if haven't gotten to 100 frames
    else {
      animation.remove(0);           // remove first frame before adding frame
      animation.add(thisFrame);      // (keeps only last 100 frames)
    }
  }
```

YORK UNIVERSITÉ UNIVERSITY

Saving from stored frames (hit target e.g.)

```
// inside moveParticle (when hit detected),
// call method to generate gif from saved frames
if (hitTarget(x, y)) {
    hit = true;
    effect.play();
    writeFramesToGif();
}
```

```
void writeFramesToGif() {

  gifExport = new GifMaker(this, "export.gif");

  gifExport.setRepeat(1);
  gifExport.setDelay(100);                            // 100ms apart

  for (int i=0; i<animation.size(); i++) {
    gifExport.addFrame(animation.get(i));
  }

  gifExport.finish();

}
```