

# Game Design

11/18/11 - Pygame



Rensselaer

# Game Architecture



Rensselaer

- import statements
- class definitions
- create initial objects
- main while loop
  - tick clock
  - process events (keypresses, etc.)
  - update objects (collision detection, etc.)
  - draw objects
  - display screen



Rensselaer

# Initialization

```
import pygame
```

```
pygame.init()
```

```
screen = pygame.display.set_mode(size)
```

- size is a tuple, common sizes are (800, 600) and (1024, 768)



Rensselaer

# Events

- `pygame.event.get()` - gets a list of all events since the last call. Call once per frame
- `event.type`:

QUIT	none
KEYDOWN	unicode, key, mod (use <code>pygame.locals</code> , for comparison)
KEYUP	key, mod
MOUSEMOTION	pos, rel, buttons
MOUSEBUTTONUP	pos, button
MOUSEBUTTONDOWN	pos, button



# Time

- `pygame.time.Clock()` Creates a Clock object
- `Clock.tick()` Returns number of milliseconds since last call
- Call once per frame to lock the frame rate.  
`Clock.tick(50) = 50FPS`
- `pygame.time.get_ticks()` Returns the number of milliseconds passed since `pygame.init()` was called



# Surfaces

- an image, stored in memory
- `surf = pygame.Surface(size)`
- `surf = pygame.image.load(filename)`
  - PNG recommended



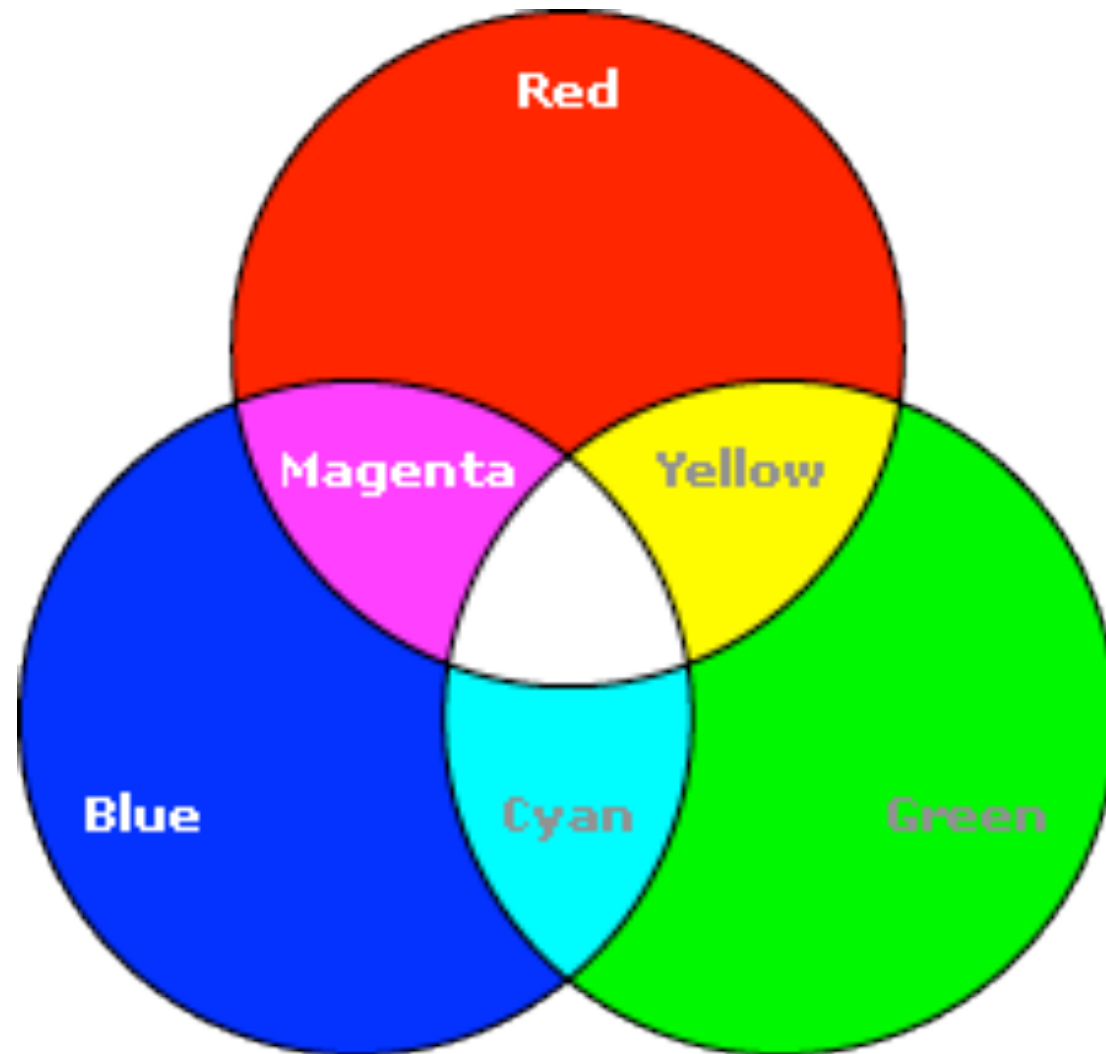
# Surface Methods

- `Surface.fill(color)` Fills surface with a solid color. Argument is a tuple of RGB values
- `Surface.convert()` Changes pixel format of the Surface's image to the format used by the main display. Makes things faster. Use it.
- `Surface.convert_alpha()` Same as above, but when the Surface's image has alpha transparency values to deal with.





Rensselaer



Rensselaer

# Blit

- From “BitBLT”, for “Bit Block Transfer”
- *copies* pixels from one surface to another

```
surface.blit( sourceSurface,  
             destPos, [ sourcePos ] )
```



Rensselaer

# More Surface Methods

- `Surface.get_rect()` Returns a Rect that will tell you the dimensions and location of the surface.
- `pygame.transform.rotate(Surface, angle)` Rotates Surface counterclockwise by degrees
- `pygame.transform.scale(Surface, (width, height))` Resizes Surface to new resolution



Rensselaer

# Homework 4 (skip!)

- Make 3 different, equally uniform-sized tiles in Photoshop or Flash
- write a program that reads in ASCII “map” files that define where the tiles go on the screen
- make the display the proper size for the tiles, and tile accordingly
- Due Friday 11/18



Rensselaer

# Tiles



land



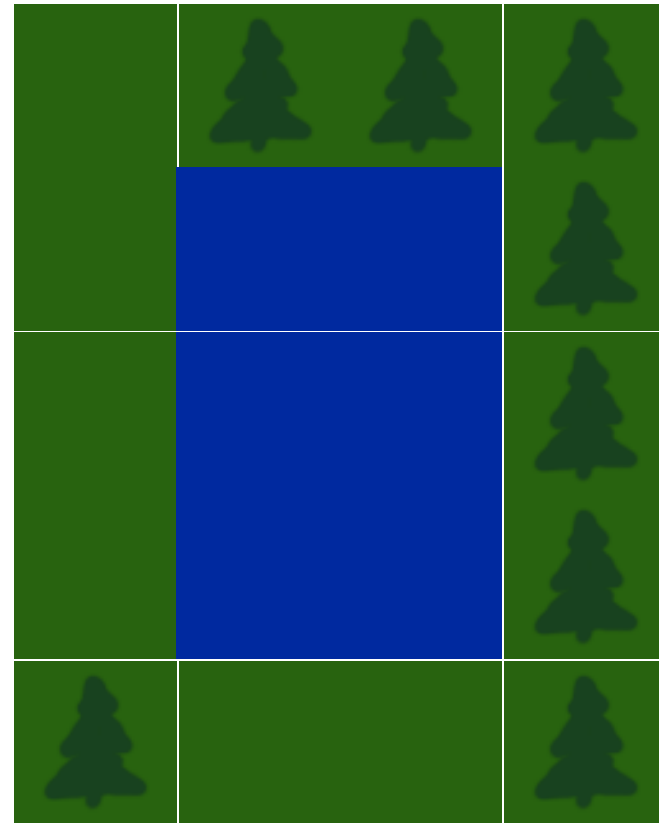
water



tree

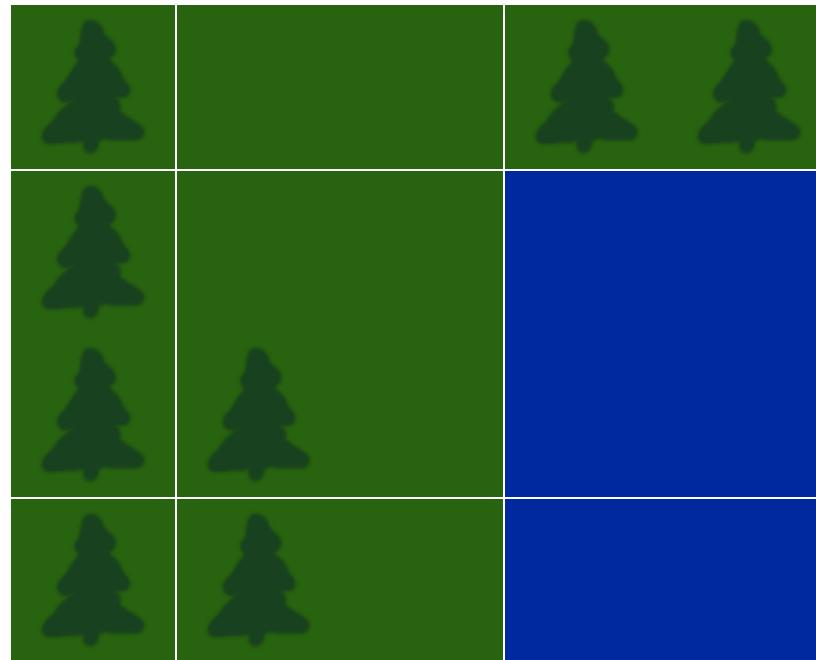


l t t t  
l w w t  
l w w t  
l w w t  
t l l t



Rensselaer

tlltt  
tllww  
ttlww  
ttlww



Rensselaer

# Homework 4 (not really)

- Make 3 different, equally uniform-sized tiles in Photoshop or Flash
- write a program that reads in ASCII “map” files that define where the tiles go on the screen
- make the display the proper size for the tiles, and tile accordingly
- Due Friday 11/18



Rensselaer

# Rects

- Position and dimension
- `Rect.move(x, y)` Returns a Rect moved `x` pixels horizontally and `y` pixels vertically  
`Rect.move_ip(x, y)` Moves the Rect `x` pixels horizontally and `y` pixels vertically

- Attributes include:

<code>top</code>	<code>left</code>	<code>bottom</code>	<code>right</code>	<code>topleft</code>	<code>bottomleft</code>
<code>topright</code>	<code>bottomright</code>	<code>midtop</code>	<code>midleft</code>	<code>midbottom</code>	<code>midright</code>
<code>center</code>	<code>centerx</code>	<code>centery</code>	<code>size</code>	<code>width</code>	<code>height</code>



Rensselaer

# Fonts

- `f = pygame.font.Font(None, 32)`
  - Creates a font object of size 32 using the default font. If you know where the .TTF file of the font you want to use is located, you can use the filename as the first argument
- `surf = f.render("Hello", 1, (255, 0, 255), (255, 255, 0))`



# Audio

```
kaboom = pygame.mixer.Sound(filename)      must be an uncompressed WAV or OGG  
kaboom.play(loops=0, maxtime=0)  
kaboom.stop()
```

```
pygame.mixer.music.load(filename)  
pygame.mixer.music.play(loops=0)          set loops to number of times to repeat after  
                                           first run-through, -1  
                                           to repeat indefinitely
```

```
pygame.mixer.music.stop()
```



Rensselaer

# Collision Detection

`Rect.contains(Rect)`: return True or False

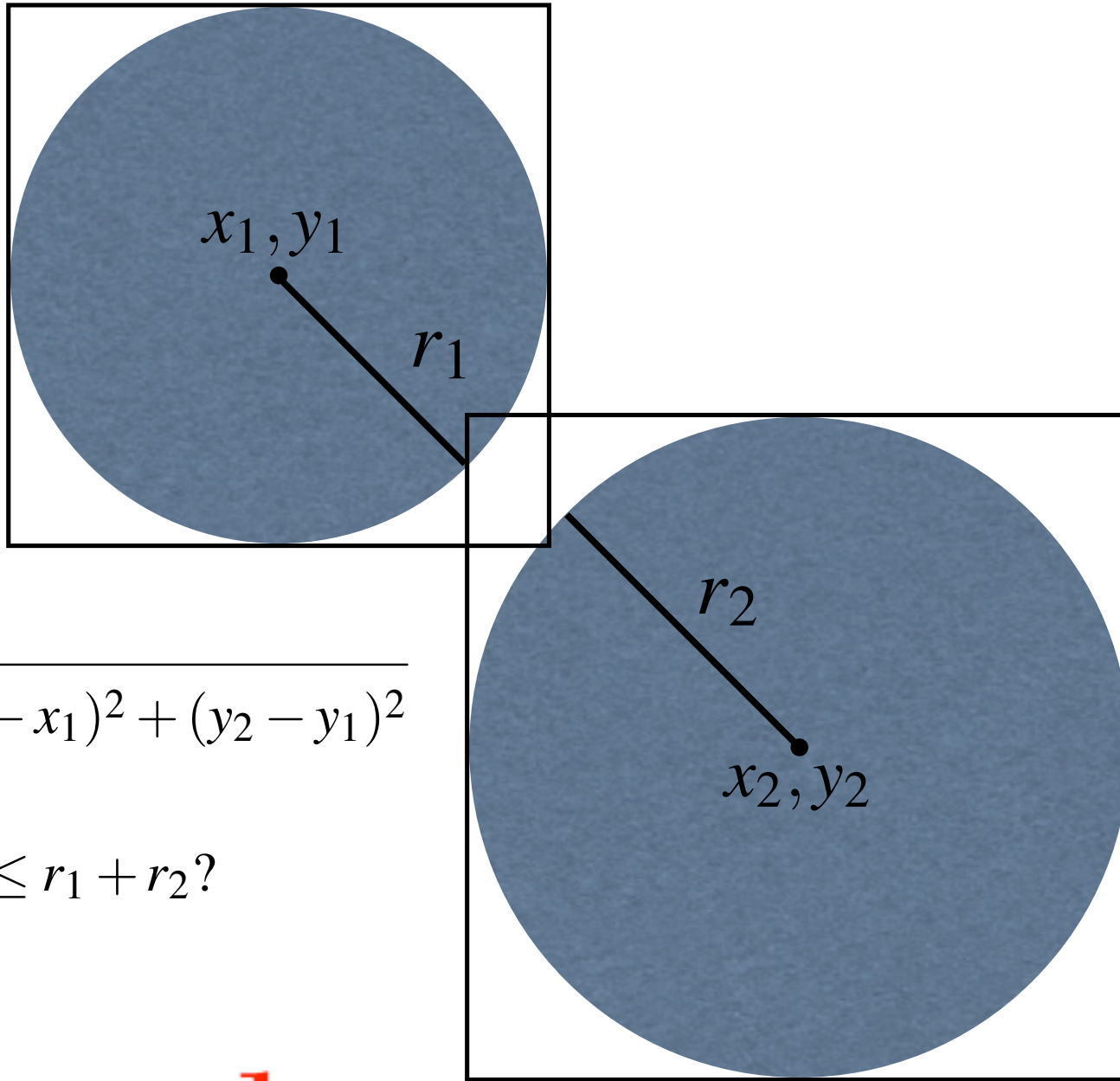
`Rect.collidepoint(x, y)`: return True or False

`Rect.colliderect(Rect)`: return True or False

`Rect.collidelist(list)`: return index (-1 if none)



Rensselaer



$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$d \leq r_1 + r_2?$$



Rensselaer

- import statements
- class definitions
- create initial objects
- main while loop
  - tick clock
  - process events (keypresses, etc.)
  - update objects (collision detection, etc.)
  - draw objects
  - `pygame.display.flip()`



# Final Assignment

- Demoed in class on 12/9 (180 seconds!)
- Choose one of two themes (to be determined momentarily)
- Tell me how you want your grade distributed among the Elemental Tetrad (Aesthetics, Mechanics, Story, Technology)
  - must allocate at least 10% to each



Rensselaer