

## Gamebox Intro

(If you haven't installed it, do so, because the rest of the class is on it)

Grades released early next week (most likely)

We use pygame to view our game (like turtle, at beginning of class)

Tychonievich wrote "gamebox", which makes it easier to use pygame

Since it's not universal, get info on gamebox pygame from:

<http://www.cs.virginia.edu/~up3f/cs1110/supplement/gamebox-overview.html>

Don't create files titled "pygame" or "gamebox", or your computer will import that file instead

Two game assignments:

One small game, submitted as homework individually

One large game, with your lab partner submitted as a final project

Example games are provided, but if you just modify those, you will not do well

<http://cs1110.cs.virginia.edu/code/gamebox/>

A blank game without a timer:

```
import pygame
import gamebox
# to make it run
camera = gamebox.Camera(800, 600)
# the screen we display to the player
def tick(keys):
    """
    This is where the game goes
    :param keys: recognizes what the user does with the arrow keys
    :return: doesn't return, is actively running
    """
    camera.display()
    # make that screen appear
ticks_per_second = 30
# controls the "lag", how many times we refresh per second
# 30 is almost always a good amount
gamebox.timer_loop(ticks_per_second, tick)
# actually runs the game
```

A blank game with a timer:

```
import pygame
import gamebox
# to make it run
camera = gamebox.Camera(800, 600)
# the screen we display to the player
time = 0
# start the timer at zero
def tick(keys):
    """
    This is where the game goes
    :param keys: recognizes what the user does with the arrow keys
    :return: doesn't return, is actively running
```

```

"""
global time
# so we can access what's outside the function
time += 1
# time goes up every time tick runs, ie 30 times per second
camera.clear("pink")
# gets rid of the last screen displayed before we display a new one
# also changes the color of our screen, I find pink relaxing
frac = str(int((time%ticks_per_second)/ticks_per_second*10))
# fractions of a second as a string
seconds = str(int((time/ticks_per_second)%60)).zfill(2)
# .zfill(2) makes it always two digits, so if we have "5" it does "05" or if we
have "0" it does "00"
minutes = str(int((time/ticks_per_second)/60))
timer = gamebox.from_text(400, 300, minutes + ":" + seconds + "." + frac,
"Arial", 24, "black")
# a function that defines a string to display on the screen
camera.draw(timer)
# displays our concatenated timer string at (400, 300) in Arial 24pt font in
black (shows well against pink)
# I put it in the middle but the example code has it at (50, 100), the top left
camera.display()
# make that screen appear
ticks_per_second = 30
# controls the "lag", how many times we refresh per second
# 30 is almost always a good amount
gamebox.timer_loop(ticks_per_second, tick)
# actually runs the game

```

## A blank game with a timer and a character:

```

import pygame
import gamebox
# to make it run
camera = gamebox.Camera(800, 600)
# the screen we display to the player
character = gamebox.from_color(camera.x, camera.y, "purple", 20, 20)
# defines a little guy at (camera.x, camera.y) adjustable coordinates in purple 20
pixels tall by 20 pixels wide
# if we don't specify, camera.x and camera.y default to the middle
time = 0
# start the timer at zero
def tick(keys):
    """
    This is where the game goes
    :param keys: recognizes what the user does with the keyboard
    :return: doesn't return, is actively running
    """
    global time
    # so we can access what's outside the function
    time += 1

```

```

# time goes up every time tick runs, ie 30 times per second
camera.clear("pink")
# gets rid of the last screen displayed before we display a new one
# also changes the color of our screen, I find pink relaxing
frac = str(int((time%ticks_per_second)/ticks_per_second*10))
# fractions of a second as a string
seconds = str(int((time/ticks_per_second)%60)).zfill(2)
# .zfill(2) makes it always two digits, so if we have "5" it does "05" or if we
have "0" it does "00"
minutes = str(int((time/ticks_per_second)/60))
timer = gamebox.from_text(400, 300, minutes + ":" + seconds + "." + frac,
"Arial", 24, "black")
# a function that defines a string to display on the screen
if pygame.K_RIGHT in keys:
    # if they press the right arrow
    character.x += 5
    # moves the guy to the right 5 pixels
if pygame.K_LEFT in keys:
    character.x -= 5
if pygame.K_UP in keys:
    # we can do it up and down, but the coordinates are reverse, idk why
    character.y -= 5
if pygame.K_DOWN in keys:
    character.y += 5
camera.draw(timer)
# displays our concatenated timer string at (400, 300) in Arial 24pt font in
black (shows well against pink)
# I put it in the middle but the example code has it at (50, 100), the top left
camera.draw(character)
camera.display()
# make that screen appear
ticks_per_second = 30
# controls the "lag", how many times we refresh per second
# 30 is almost always a good amount
gamebox.timer_loop(ticks_per_second, tick)
# actually runs the game

```

Fiddle with these, try to make some games and figure out how the example games work

Have fun playing with gamebox and creating some game !