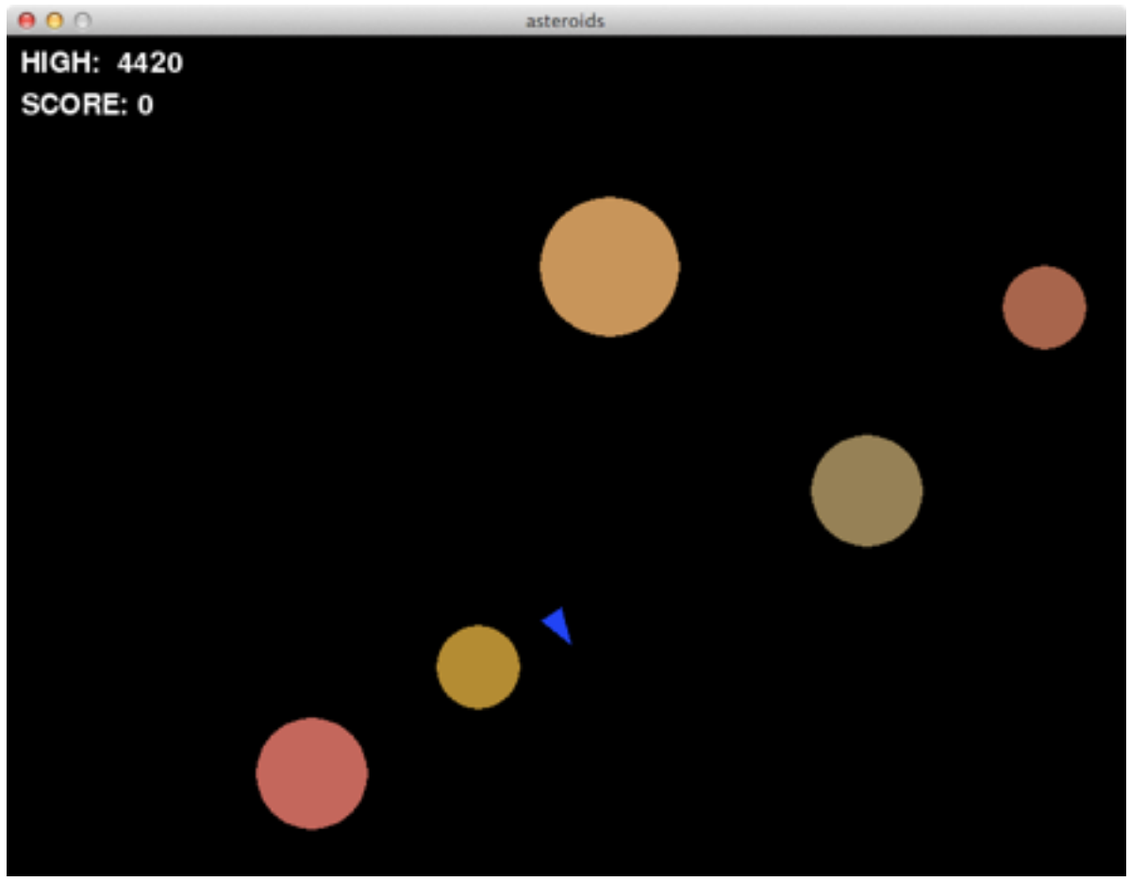
COMP10001 Foundations of Computing
Semester 2 2014
Lecture 23 (advanced lecture, not examinable)

Computer games in Python

Bernie Pope, bjpope@unimelb.edu.au

Outline

- Asteroids.
- The `pygame` library.
- Vectors.
- Game objects.



Asteroids

- Fly a space ship through an asteroid field.
- Shoot the asteroids with your gun.
- Don’t crash into the asteroids!
- Ship can rotate on the spot.
- Acceleration is only in the forwards facing direction.

pygame

- Open source computer game library in Python.
- http://www.pygame.org/
- Not part of the Python standard library: you must install it on your own computer. Does not work on IVLE.
- Book: Making games with Python & Pygame:

  http://inventwithpython.com/pygame/

SDL

- Pygame is built on top of SDL (Simple DirectMedia Layer):
  - Provides portable access to input and output devices (screen, audio, keyboard, mouse, joystick).
  - Works on Windows, OS X, Linus, iOS, Android.
  - http://www.libsdl.org/

pygame basics

```
pygame.init()

screen = pygame.display.set_mode((MAX_X, MAX_Y), 0, 32)
```

pygame basics

Initialise pygame. You must do this first.

```
pygame.init()
```

```
screen = pygame.display.set_mode((MAX_X, MAX_Y), 0, 32)
```

pygame basics

Create a screen to display the game graphics.

```
pygame.init()
screen = pygame.display.set_mode((MAX_X, MAX_Y), 0, 32)
```

Arguments are width, height, flags and colour depth.

pygame basics

Fill the background of the screen with black pixels.

```
BLACK = (0, 0, 0)

screen.fill(BLACK)
```

pygame basics

```
from random import randint


def random_colour():
    red = randint(0, 255)
    green = randint(0, 255)
    blue = randint(0, 255)
    return (red, green, blue)
```

A function to generate a random (R, G, B) colour.

pygame basics

```
def random_circle(surface):

    x = randint(0, MAX_X - 1)

    y = randint(0, MAX_Y - 1)

    radius = randint(10, 30)

    colour = random_colour()

    pygame.draw.circle(surface, colour, (x, y), radius)
```

A function to draw a circle with random centre position, random radius and random colour.

pygame basics

```
def game_loop(screen):
    while True:
        for event in pygame.event.get():
            if event.type == KEYDOWN:
                if event.key == K_SPACE:
                    random_circle(screen)
            elif event.type == QUIT:
                pygame.quit()
                sys.exit()
    pygame.display.update()
```

pygame basics

```
def game_loop(screen):
    while True:
        for event in pygame.event.get():
            if event.type == KEYDOWN:
                if event.key == K_SPACE:
                    random_circle(screen)
            elif event.type == QUIT:
                pygame.quit()
                sys.exit()
    pygame.display.update()
```

Loop forever (until the player quits).

pygame basics

```
def game_loop(screen):
    while True:
        for event in pygame.event.get():
            if event.type == KEYDOWN:
                if event.key == K_SPACE:
                    random_circle(screen)
            elif event.type == QUIT:
                pygame.quit()
                sys.exit()
    pygame.display.update()
```

Process the next game events, such as key presses.

pygame basics

```
def game_loop(screen):
    while True:
        for event in pygame.event.get():
            if event.type == KEYDOWN:
                if event.key == K_SPACE:
                    random_circle(screen)
        elif event.type == QUIT:
            pygame.quit()
            sys.exit()
    pygame.display.update()
```

If the space bar was pressed, draw a random circle on the screen.

pygame basics

```
def game_loop(screen):
    while True:
        for event in pygame.event.get():
            if event.type == KEYDOWN:
                if event.key == K_SPACE:
                    random_circle(screen)
            elif event.type == QUIT:
                pygame.quit()
                sys.exit()
        pygame.display.update()
```

If the user closed the window then shut down the game and exit the program.

pygame basics

```
def game_loop(screen):
    while True:
        for event in pygame.event.get():
            if event.type == KEYDOWN:
                if event.key == K_SPACE:
                    random_circle(screen)
            elif event.type == QUIT:
                pygame.quit()
                sys.exit()
    pygame.display.update()
```

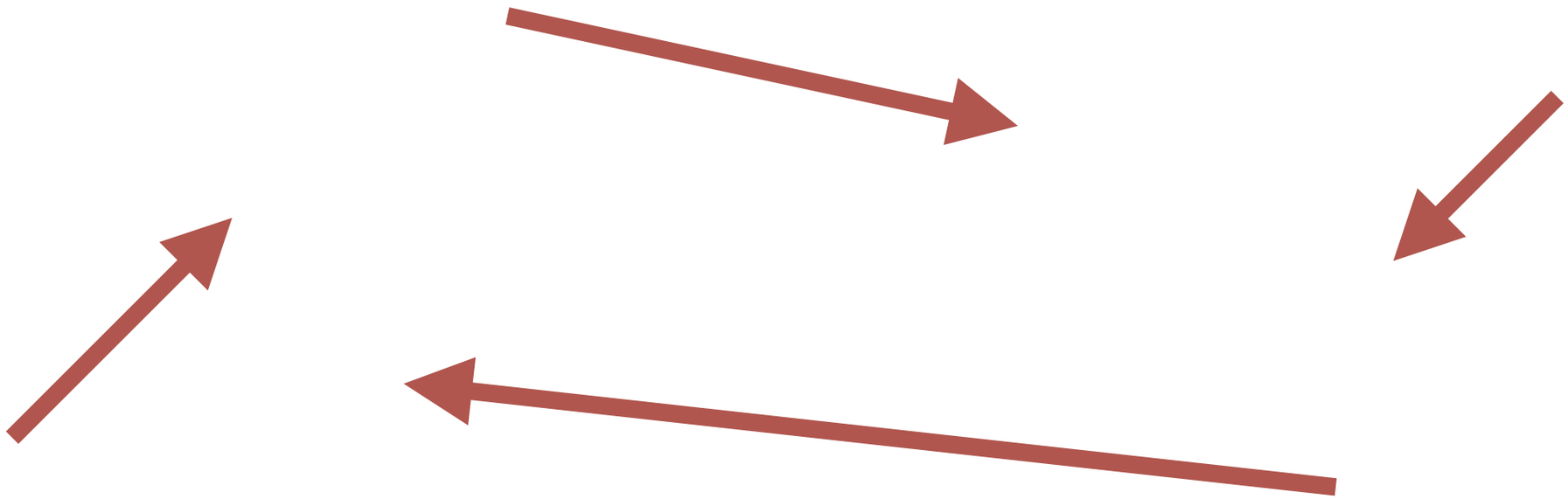
Redraw the screen to make all changes visible.

Vectors

- Vectors are mathematical objects that have:
  - A direction.
  - A magnitude.

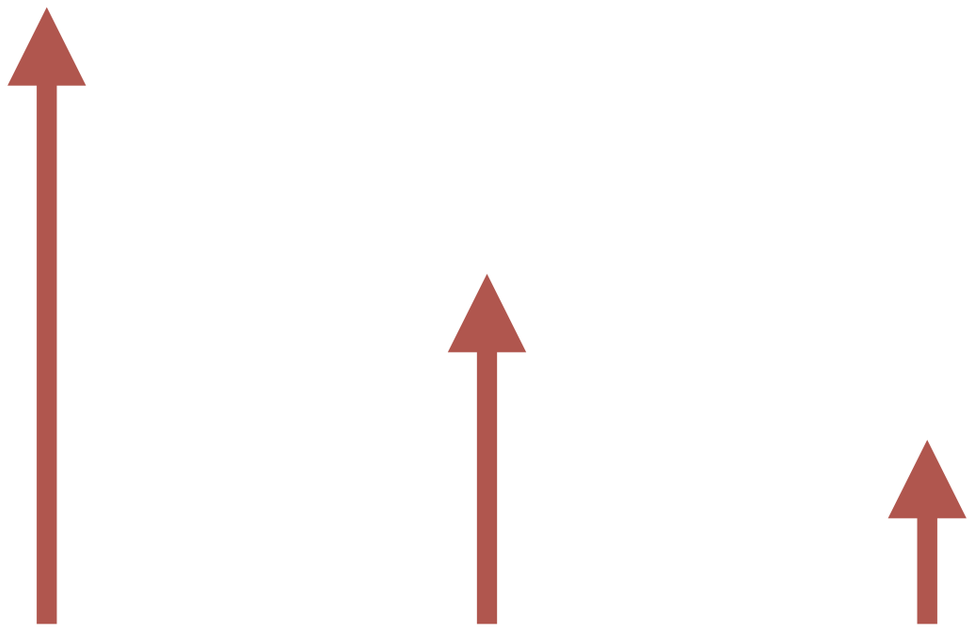
Vectors

- You can visualise a two dimension vector as an arrow.
- An arrow points in a direction and has a length.



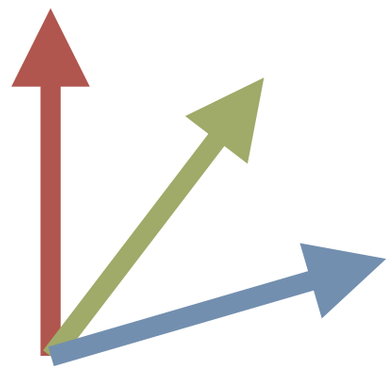
Vectors

- Vectors can be scaled to make them bigger or smaller.



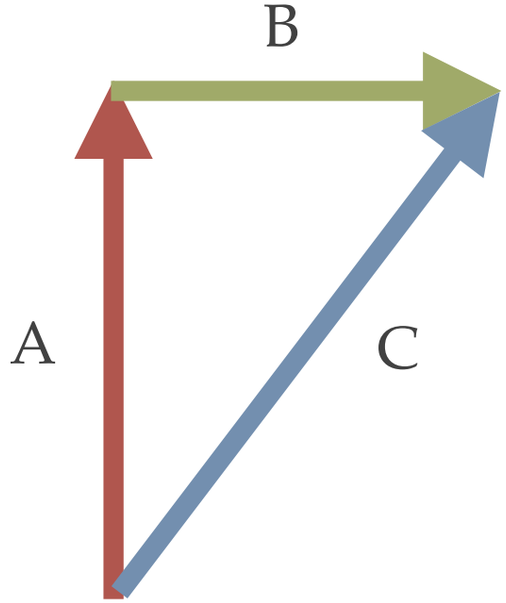
Vectors

- Vectors can be rotated to change their direction.



Vectors

- Vectors can be added together to give new vectors.



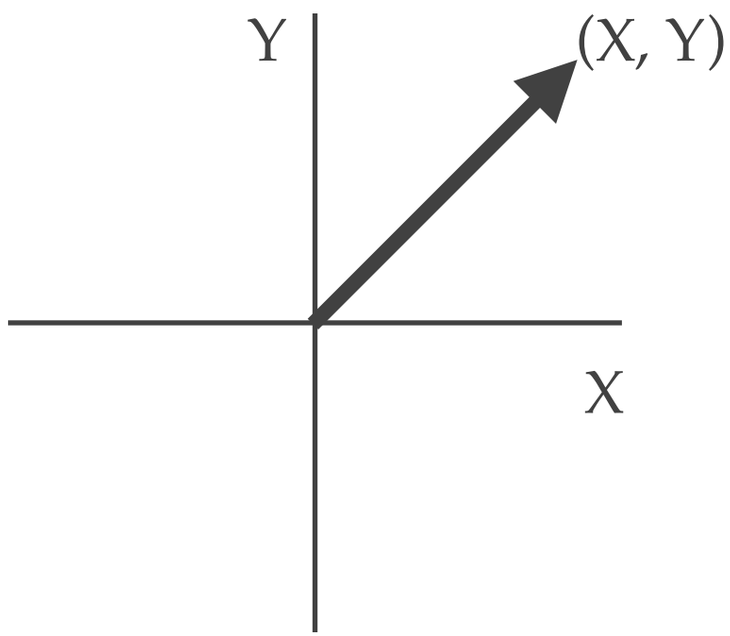
$A + B = C$

Motion of objects in a game

- Different objects (space ship, rocks, bullets) in the asteroids game move about over time.
- We can describe their motion using vectors:
  - Their position is a 2D vector.
  - Their velocity is a 2D vector.

Position as a vector

- The position of a game object is its coordinates in 2D dimensions.
- We can think of that as a vector starting at the origin and ending at the object position.



Velocity

- Each object has a velocity:
  - the direction in which it is travelling.
  - the speed at which it is moving.

Velocity as a vector

- We can represent each object's velocity as a vector:
  - Direction of the object is the direction of the vector.
  - Speed of the object is the magnitude of the vector.

Motion of objects in a game

- At each time step in the game we compute the new position of each object like so:

```
new_position = old_position + velocity
```

A vector module

```
>>> from vector2D import Vec2d

>>> v1 = Vec2d(3, 7)

>>> v2 = Vec2d(12, -2)

>>> v3 = v1 + v2

>>> v3

Vec2d(15, 5)
```

Asteroids uses the vector2D module from the pygame website.

Game objects

- Games tend to have lots of dynamic entities:
  - Space ships.
  - Rocks.
  - Bullets.

Game objects

- Each entity has its own internal state, e.g.:
    - Position.
    - Velocity.
    - Health.
    - Size.
    - Colour.

Game objects

- Each entity has its own behaviours, e.g.:
  - Move.
  - Rotate.
  - Explode.
  - Respawn.

Game objects

- Some aspects of state and behaviour are shared between entities, and some are distinct.

Game objects

- We can use Python's classes to help us:
  - Share common aspects of state and behaviour.
  - Add new state and behaviour where needed.

Game objects

```
class GameObject(object):
    def __init__(self, position, velocity):
        self.position = position
        self.velocity = velocity

    def move(self):
        self.position = self.position + self.velocity
        if self.position.x >= MAX_X:
            self.position.x = 0
        elif self.position.x < 0:
            self.position.x = MAX_X - 1
        if self.position.y >= MAX_Y:
            self.position.y = 0
        elif self.position.y < 0:
           self.position.y = MAX_Y - 1
```

All game objects in asteroids have a position and a velocity. And they all move in the same way.

Game objects

```
class Rock(GameObject):

    def __init__(self, position, velocity, radius, colour):

        super(Rock, self).__init__(position, velocity)

        self.radius = radius

        self.colour = colour


def draw(self, windowSurface):

    center = (int(self.position.x), int(self.position.y))

    pygame.draw.circle(windowSurface, self.colour, center, self.radius)
```

Rocks are kinds of GameObjects that have their own way of drawing.

Game objects

- `SpaceShips` are `GameObjects` which can accelerate and rotate.
- `Bullets` are `GameObjects` which have a limited lifespan.

Conclusion

- Writing games is a lot of fun, and a great way to learn programming.
- Making fun games is a LOT of work.
- Real games are usually made by teams of people, which also include graphic artists and musicians.
- Feel free to take the `asteroids.py` code and add your own features.