

2D 게임 프로그래밍

- 반드시 카메라를 ON 하고 !
- 입장 이름은 "학번 이름"으로 설정 !
- 미리 수업 git 서버에서 자료를 Pull 해서 준비 !

Lecture #11. 캐릭터 컨트롤러

2D 게임 프로그래밍

이대현 교수

학습 내용

- 클래스 변수

- 캐릭터 컨트롤러

- 상태기계

- 이벤트 큐



1000명 선수
리소스 로딩 최적화

boys_team_1000.py

초기 로딩하는 시간이
많이 걸린다?



문제점은?

```
class Boy:

    def __init__(self):
        self.x, self.y = random.randint(100, 700), 90
        self.frame = random.randint(0, 7)
        self.image = load_image('run_animation.png')
```

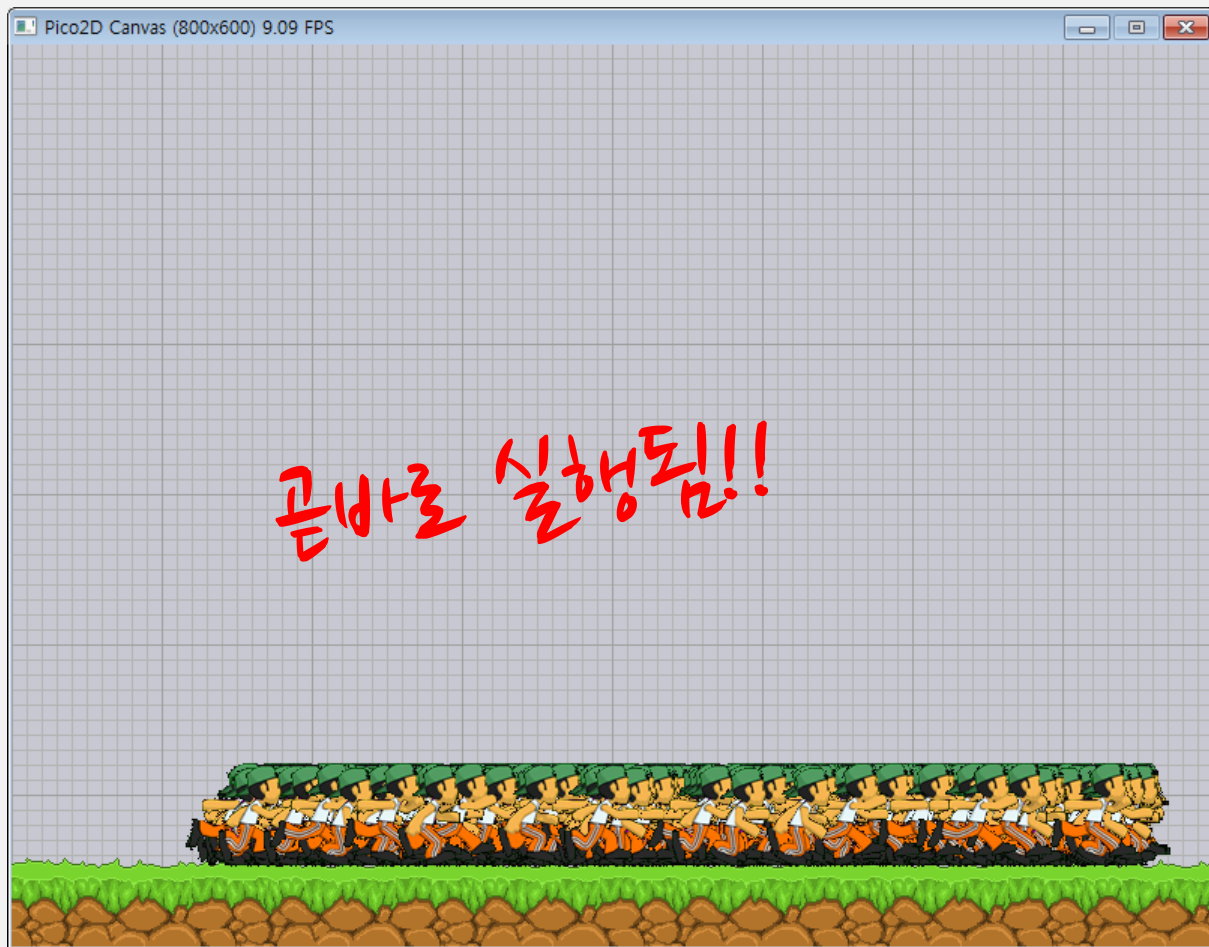
객체의 멤버변수는 객체마다 따로 만들어진다!

1000번의 로딩이 반복



```
class Boy:
    image = None

    def __init__(self):
        self.x, self.y = random.randint(100, 700), 90
        self.frame = random.randint(0, 7)
        if Boy.image == None:
            Boy.image = load_image('run_animation.png')
```



클래스 변수

클래스 자체에 할당되는 변수.
객체들은 공유하는 동일한 변수를 갖게 됨.

```
class Boy:  
    image = None  
  
...  
... def __do_some():  
...  
    Boy.image = ...
```

```
class Boy:
    image = None

    def __init__(self):
        self.x, self.y = random.randint(100, 700), 90
        self.frame = random.randint(0, 7)
        if Boy.image == None:
            Boy.image = load_image('run_animation.png')
```

단 한번의 이미지 로딩만 수행.
이미지 리소스를 모든 객체가 공유하게 됨.

self 없는 클래스

```
class Star:

    type = 'Star'
    x = 100

    def change():
        x = 200
        print('x is ', x)

print('x IS ', Star.x) # OK
Star.change() # OK
print('x IS ', Star.x)

star = Star() # OK
print('x IS ', star.x) # OK
star.change() # Error
```

--init--
오류

self 의 의미

```
class Player:
    type = 'Player'

    def __init__(self):
        self.x = 100

    def where(self):
        print(self.x)

player = Player()
player.where()

# 클래스 변수 사용
print(Player.type)

# 클래스 함수 호출
Player.where() # error
Player.where(player) # OK, player.where() 과 같음.
```

캐릭터 컨트롤러(Character Controller)

■ 게임 주인공의 행동을 구현한 것!

- 키입력에 따른 액션
- 주변 객체와의 인터랙션

■ 게임 구현에서 가장 핵심적인 부분임.



우리의 “주인공”은?

■캐릭터 컨트롤러의 행위를 적으면...

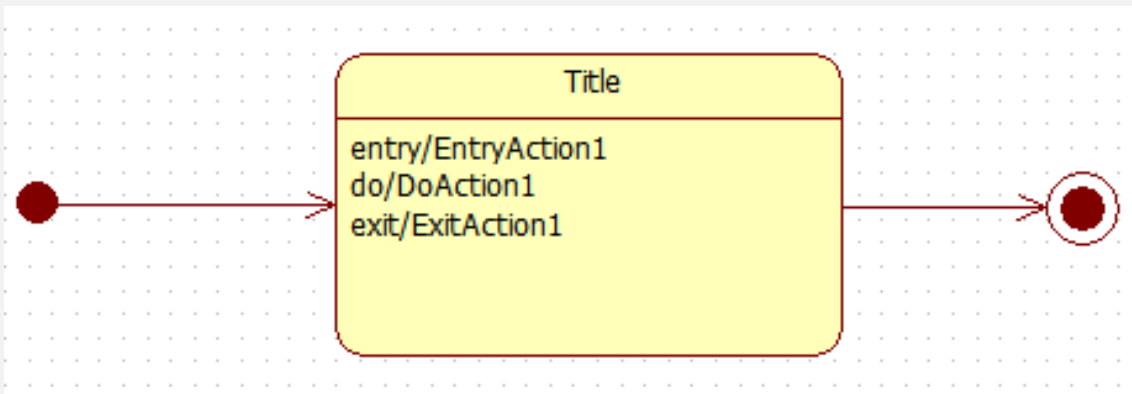
- 처음 소년의 상태는 제자리에 서서 휴식을 하고 있습니다.
- 이 상태에서 오른쪽 방향키를 누르면 소년은 오른쪽으로 달리게 됩니다.
- 방향키를 계속 누르고 있으면, 소년도 계속 오른쪽으로 달리죠.
- 방향키에서 손가락을 떼면 소년은 달리기를 멈추고 휴식상태에 들어갑니다.
- 한참 지나도, 방향키 입력이 없으면 소년은 취침에 들어갑니다.
- 달리는 중에, Dash 키를 누르면 빠르게 달립니다.
- 왼쪽 방향키 조작에 대해선 왼쪽으로 달리게 됩니다.
- 캔버스의 좌우측 가장자리에 도착하면 더 이상 달려나가지는 않습니다.

상태 다이어그램(State Diagram)

- 시스템의 변화를 모델링하는 다이어그램.
- 사건이나 시간에 따라 시스템 내의 객체들이 자신의 상태(state)를 바꾸는 과정을 모델링함.
- 모델링, 명세, 그리고 구현에 모두 사용되는 강력한 툴
- 상태(state)의 변화 예
 - 스위치를 누를 때마다 탁상 전등 상태는 “켜짐”에서 “꺼짐”으로 바뀐다.
 - 리모트 컨트롤의 버튼을 누르면 TV의 상태는 한 채널을 보여주다가 다른 상태를 보여주게 된다.
 - 얼마간의 시간이 흐르면 세탁기의 상태는 “세탁”에서 “헹굼”으로 바뀐다.

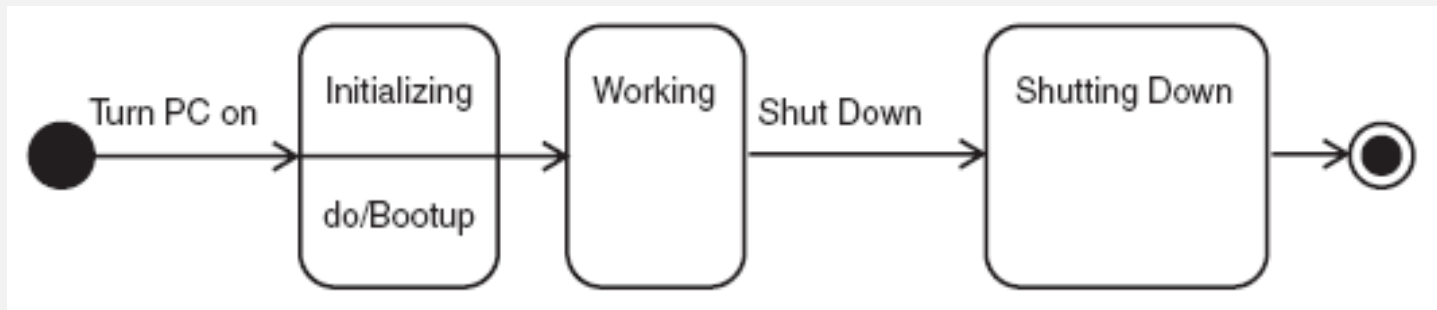
상태(State)

- 상태 : 어떤 조건을 만족하는 동안 머무르면서, 정해진 일을 수행하고 이벤트를 기다리는 “상황”
- Entry action : 특정한 상태로 들어갈 때마다 발생하는 일
- Exit action : 특정한 상태에서 나갈 때마다 발생하는 일
- Do activity : 특정 상태에 머무르는 동안 수행하는 일(반복될 수 있음)



상태 변화(State Transition)

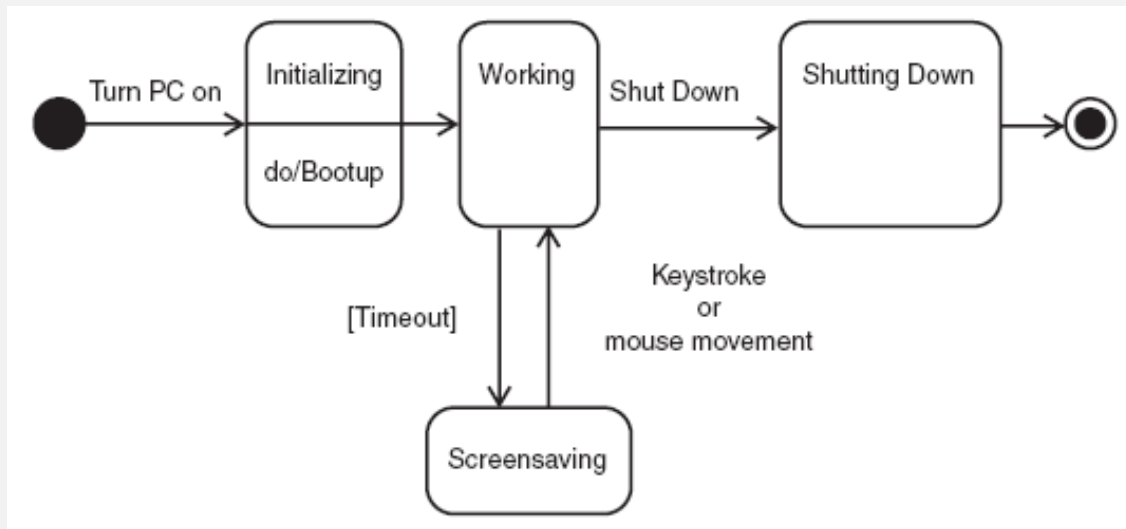
- A transition is a relationship between two states; it indicates that an object in the first state will perform certain actions, then enter the second state when a given event occurs.



이벤트(Event)

■ 상태 변화(State Transition)을 일으키는 원인이 되는 일

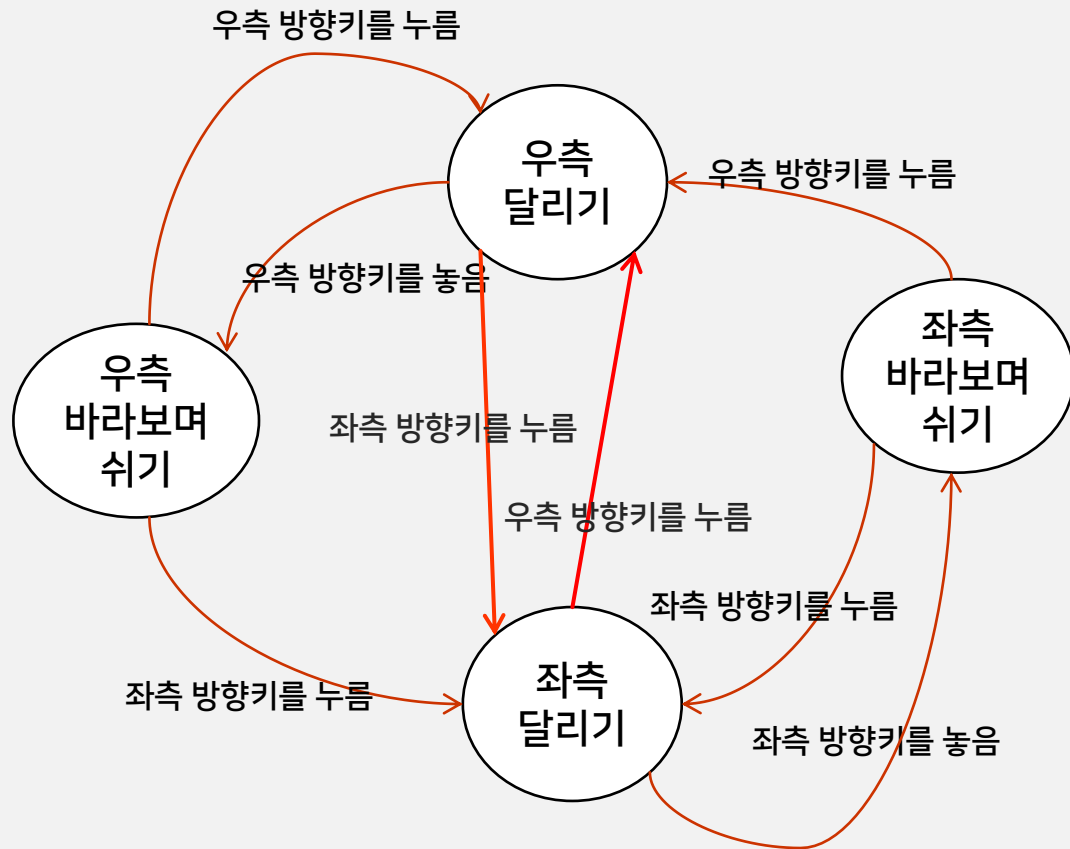
- 외부적인 이벤트 : 예) 키보드 입력
- 내부적인 이벤트 : 예) 타이머
- 경우에 따라서는 이벤트 없이도 상태 변화가 있을 수 있음.



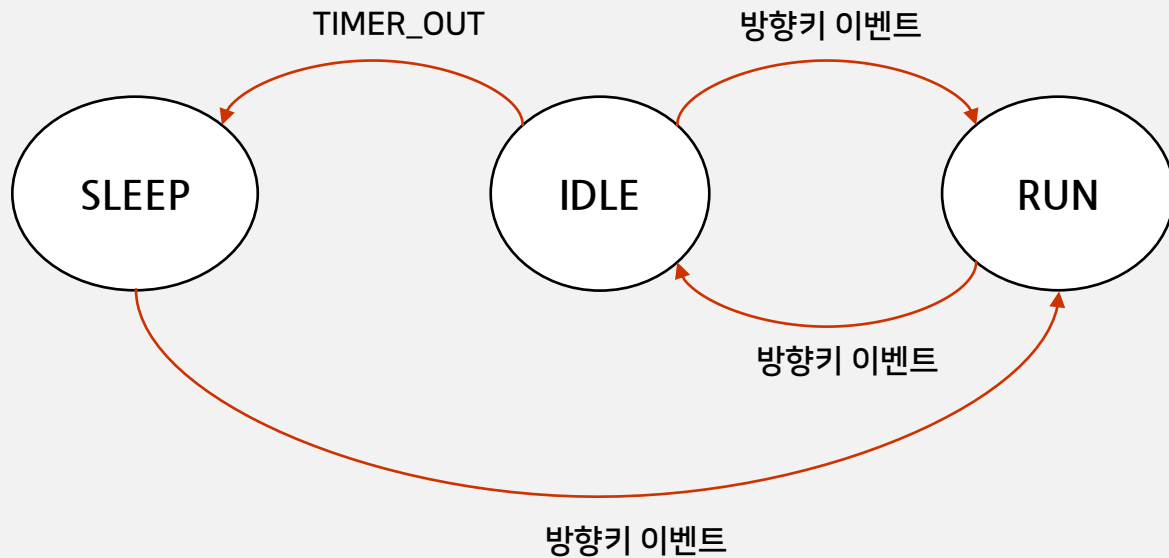
상태와 이벤트 찾기

- 주인공의 움직임 상태를 찾아보자.
- 주인공의 상태에 변화를 일으킬 수 있는 이벤트를 찾아보자.

상태 다이어그램 #1



상태 다이어그램 #2





캐릭터 컨트롤러
구현(IDLE & RUN)

Source Code Files

- `game_framework.py`
- `mygame.py` – 실행 시작 파일
- `main_state.py` – 메인 게임상태
- `Grass.py` – 잔디 클래스
- `Boy.py` – 소년 클래스 (실습 코딩)



```
# Boy Event
RIGHT_DOWN, LEFT_DOWN, RIGHT_UP, LEFT_UP = range(4)

key_event_table = {
    (SDL_KEYDOWN, SDLK_RIGHT): RIGHT_DOWN,
    (SDL_KEYDOWN, SDLK_LEFT): LEFT_DOWN,
    (SDL_KEYUP, SDLK_RIGHT): RIGHT_UP,
    (SDL_KEYUP, SDLK_LEFT): LEFT_UP
}
```


boy.py – Idle State



```
class IdleState:
    def enter(boy, event):
        if event == RIGHT_DOWN:
            boy.velocity += 1
        elif event == LEFT_DOWN:
            boy.velocity -= 1
        elif event == RIGHT_UP:
            boy.velocity -= 1
        elif event == LEFT_UP:
            boy.velocity += 1
        boy.timer = 1000

    def exit(boy, event):
        pass

    def do(boy):
        boy.frame = (boy.frame + 1) % 8
        boy.timer -= 1

    def draw(boy):
        if boy.dir == 1:
            boy.image.clip_draw(boy.frame * 100, 300, 100, 100, boy.x, boy.y)
        else:
            boy.image.clip_draw(boy.frame * 100, 200, 100, 100, boy.x, boy.y)
```

boy.py – Run State



```
class RunState:
    def enter(boy, event):
        if event == RIGHT_DOWN:
            boy.velocity += 1
        elif event == LEFT_DOWN:
            boy.velocity -= 1
        elif event == RIGHT_UP:
            boy.velocity -= 1
        elif event == LEFT_UP:
            boy.velocity += 1
        boy.dir = boy.velocity

    def exit(boy, event):
        pass

    def do(boy):
        boy.frame = (boy.frame + 1) % 8
        boy.timer -= 1
        boy.x += boy.velocity
        boy.x = clamp(25, boy.x, 800 - 25)

    def draw(boy):
        if boy.velocity == 1:
            boy.image.clip_draw(boy.frame * 100, 100, 100, 100, boy.x, boy.y)
        else:
            boy.image.clip_draw(boy.frame * 100, 0, 100, 100, boy.x, boy.y)
```



```
next_state_table = {  
    IdleState: {RIGHT_UP: RunState, LEFT_UP: RunState,  
                RIGHT_DOWN: RunState, LEFT_DOWN: RunState},  
    RunState: {RIGHT_UP: IdleState, LEFT_UP: IdleState,  
               LEFT_DOWN: IdleState, RIGHT_DOWN: IdleState}  
}
```



```
class Boy:

    def __init__(self):
        self.x, self.y = 800 // 2, 90
        self.image = load_image('animation_sheet.png')
        self.dir = 1
        self.velocity = 0
        self.frame = 0
        self.timer = 0
        self.event_que = []
        self.cur_state = IdleState
        self.cur_state.enter(self, None)
```



```
def add_event(self, event):
    self.event_que.insert(0, event)

def update(self):
    self.cur_state.do(self)
    if len(self.event_que) > 0:
        event = self.event_que.pop()
        self.cur_state.exit(self, event)
        self.cur_state = next_state_table[self.cur_state][event]
        self.cur_state.enter(self, event)

def draw(self):
    self.cur_state.draw(self)

def handle_event(self, event):
    if (event.type, event.key) in key_event_table:
        key_event = key_event_table[(event.type, event.key)]
        self.add_event(key_event)
```

시
스
립



캐릭터 컨트롤러 구현

(IDLE & RUN & SLEEP)

boy.py – SLEEP_TIMER 이벤트 추가



```
# Boy Event
```

```
RIGHT_DOWN, LEFT_DOWN, RIGHT_UP, LEFT_UP, SLEEP_TIMER = range(5)
```

```
key_event_table = {  
    (SDL_KEYDOWN, SDLK_RIGHT): RIGHT_DOWN,  
    (SDL_KEYDOWN, SDLK_LEFT): LEFT_DOWN,  
    (SDL_KEYUP, SDLK_RIGHT): RIGHT_UP,  
    (SDL_KEYUP, SDLK_LEFT): LEFT_UP  
}
```

boy.py – SLEEP 상태 함수 추가



```
class SleepState:

    def enter(boy, event):
        boy.frame = 0

    def exit(boy, event):
        pass

    def do(boy):
        boy.frame = (boy.frame + 1) % 8

    def draw(boy):
        if boy.dir == 1:
            boy.image.clip_composite_draw(boy.frame * 100, 300, 100, 100,
                                           3.141592 / 2, '', boy.x - 25, boy.y - 25, 100, 100)
        else:
            boy.image.clip_composite_draw(boy.frame * 100, 200, 100, 100,
                                           -3.141592 / 2, '', boy.x + 25, boy.y - 25, 100, 100)
```


boy.py – Sleep 상태 변화 추가



```
next_state_table = {
    IdleState: {RIGHT_UP: RunState, LEFT_UP: RunState,
                RIGHT_DOWN: RunState, LEFT_DOWN: RunState,
                SLEEP_TIMER: SleepState},
    RunState: {RIGHT_UP: IdleState, LEFT_UP: IdleState,
              LEFT_DOWN: IdleState, RIGHT_DOWN: IdleState},
    SleepState: {LEFT_DOWN: RunState, RIGHT_DOWN: RunState,
                LEFT_UP: RunState, RIGHT_UP: RunState}
}
```

boy.py – IdleState SLEEP_TIMER 이벤트 처리



```
class IdleState:

    def do(boy):
        boy.frame = (boy.frame + 1) % 8
        boy.timer -= 1
        if boy.timer == 0:
            boy.add_event(SLEEP_TIMER)
```

`clip_composite_draw(left, bottom, width, height, rad, flip, x, y, w, h)`

rad: 회전각도(라디안값)

flip: 반전여부('h': 상하반전, 'v':좌우반전, 'hv': 상하좌우반전)

Pico2d debug_print

```
def draw(self):  
    self.cur_state.draw(self)  
    debug_print('Velocity :' + str(self.velocity) + ' Dir:' + str(self.dir))
```