



Gra BlobbyVolley w języku Python



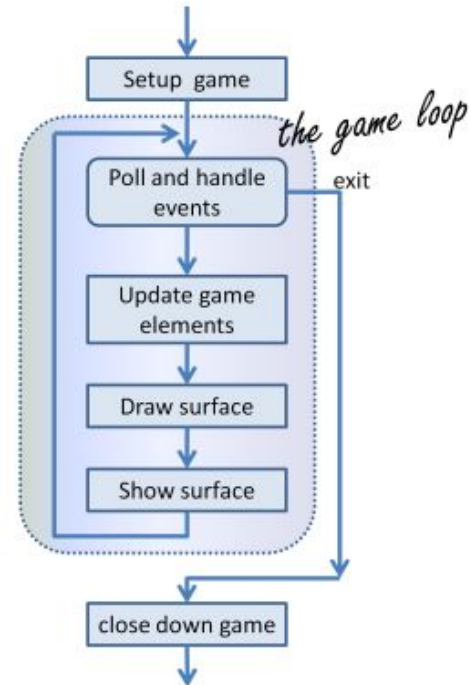
Krótko o pygame

- prosty w użyciu
- najpopularniejsza bibliotek wspomagająca tworzenie gier w Python
- duża społeczność
- powolny w porównaniu do np. Unity

Podstawowe funkcje - inicjalizacja

`pygame.init()` oraz `pygame.quit()`

Inicjalizacja / wyłączenie modułów





Podstawowe funkcje - wyświetlanie

```
pygame.display.set_mode()  
window = pygame.display.set_mode((640, 480))  
window = pygame.display.set_mode((0, 0), pygame.FULLSCREEN)
```

```
pygame.transform.scale()  
bg = pygame.image.load("tlo.png")  
bg = pygame.transform.scale(bg, (640, 480))'
```

```
pygame.Surface.blit()  
window.blit(bg, (0, 0))
```



Podstawowe funkcje - rysowanie

```
pygame.draw.line()
```

```
pygame.draw.line(window, (255, 255, 0), (x1, y1), (x2, y2))
```

```
pygame.draw.circle()
```

```
pygame.draw.circle(window, (255, 255, 0), (x, y), radius)
```

```
pygame.draw.polygon()
```

```
pygame.draw.polygon(window, (255, 255, 0), ((x1, y1), (x2, y2), (x3, y3), (x4, y4), (x5, y5), (x6, y6)))
```



Podstawowe funkcje - input

```
pygame.key.get_pressed()  
keys = pygame.key.get_pressed()
```

```
if keys[pygame.K_UP]:  
    # Move forward  
if keys[pygame.K_RIGHT]:  
    # Move right
```

```
pygame.event.get()
```



Pygame - przykład prostego programu

<https://github.com/JakubBekier/BlobbyVolley/tree/master/Zadania/zadanie2>

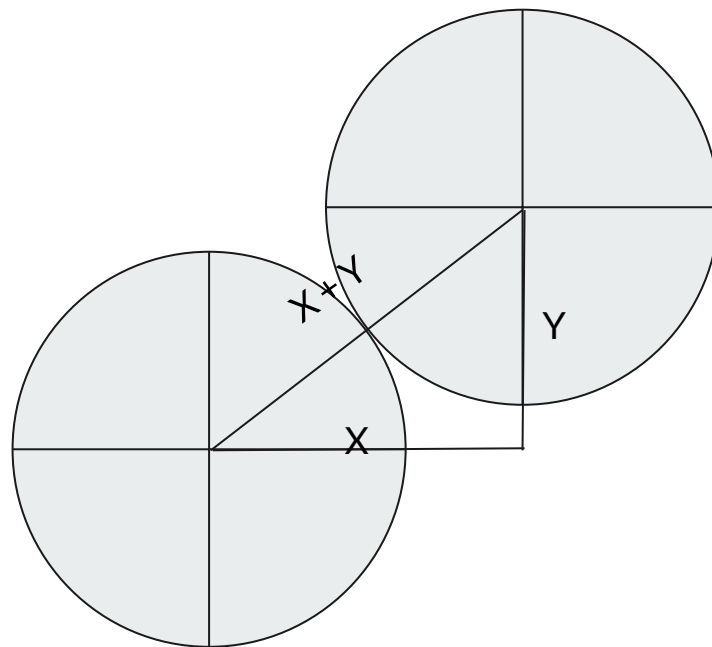


Struktura aplikacji

<https://github.com/JakubBekier/BlobbyVolley/tree/master/BlobbyVolleyGame>

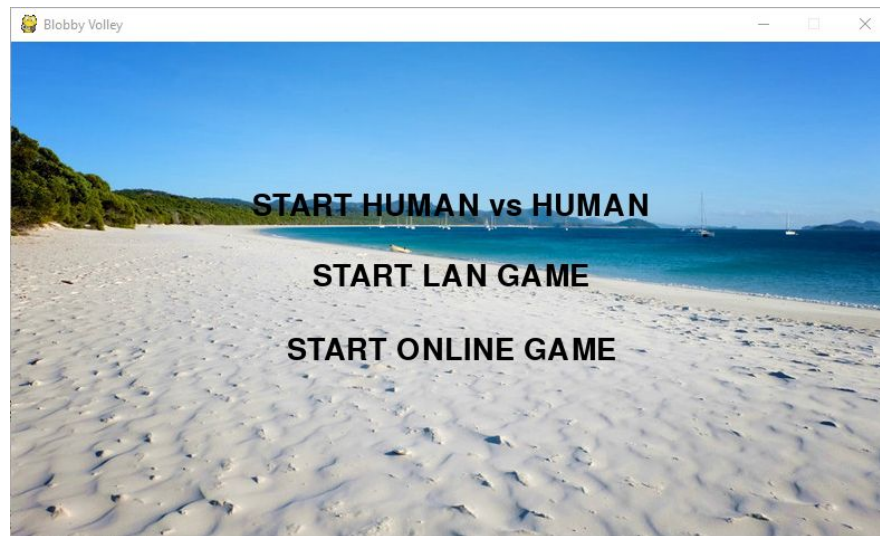
Fizyka piłki

Piłce nadawana jest prędkość w osi poziomej i pionowej, zależy od kąta, pod którym zderzyła się z graczem.



Gra przez sieć

- dostępne są dwie opcje gry, przez sieć LAN i przez Internet
- obie opcje wykorzystują serwer do komunikacji



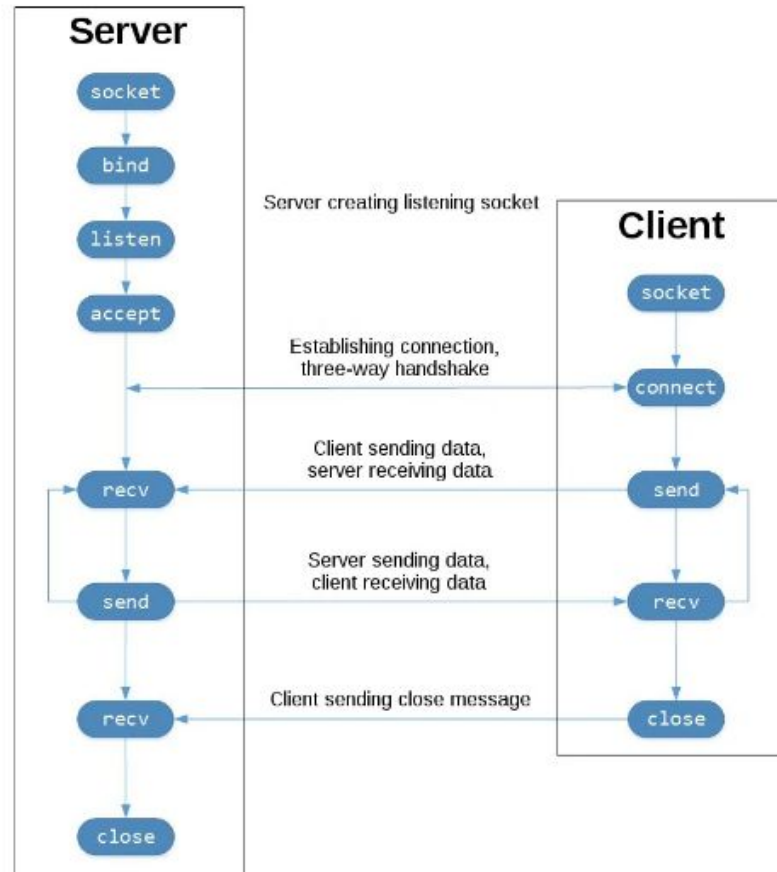


Dlaczego korzystamy z serwerów?

- bezpieczeństwo
- prostota nawiązywania połączenia
- kontrola i rozwiązywanie konfliktów przez serwer
- stabilność

Ogólny model połączenia klient-serwer

- połączenie oparte na protokole TCP
- socket - wybór typu gniazda
- bind - przypisanie adresu i portu
- listen - przełączenie gniazda w tryb nasłuchu
- accept - uzyskanie nowego socketu do konkretnego klienta, domyślnie blokująca funkcja





Praktyczny przykład

```
import socket
from _thread import *

server = '127.0.0.1'
port = 5555
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((server, port))
s.listen()

def threaded_client(conn, ...):
    pass

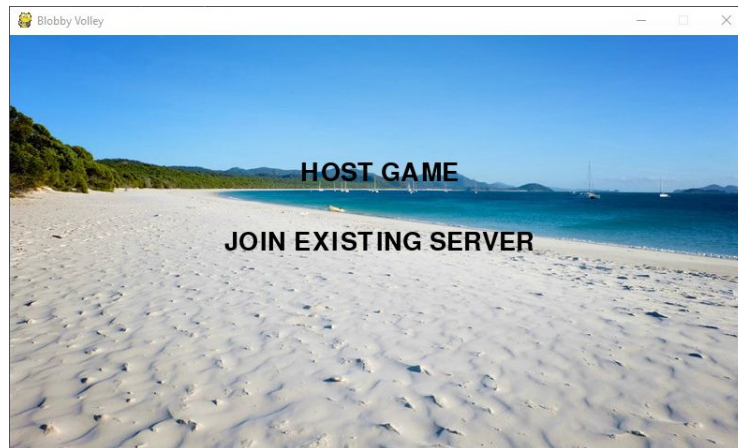
while True:
    conn, addr = s.accept()
    start_new_thread(threaded_client, (conn, ...))
```

```
import socket

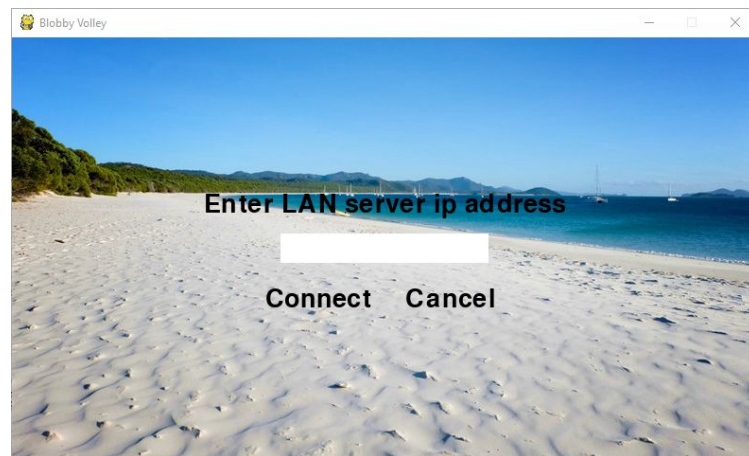
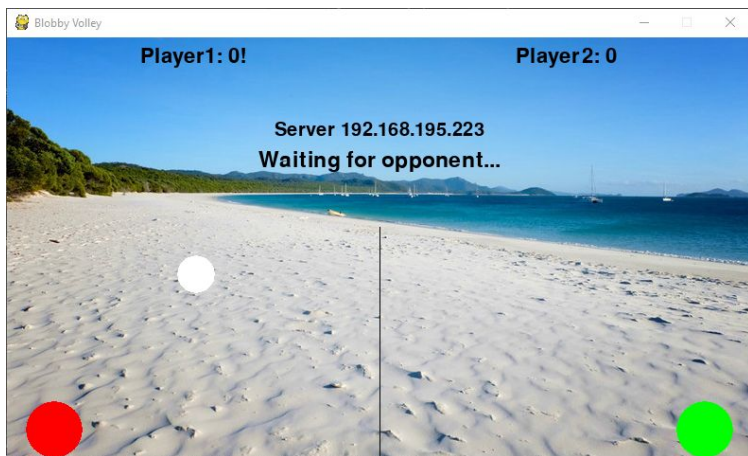
server = '127.0.0.1'
port = 5555
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((server, port))
s.send(...)
data = s.recv(2048)
s.close()
```

Tryb gry przez LAN w naszej aplikacji

- HOST GAME - w oddzielnym wątku uruchamiamy serwer, a następnie do niego dołączamy
- JOIN EXISTING SERVER - wyświetlane jest okno z prośbą o wpisanie adresu ip serwera

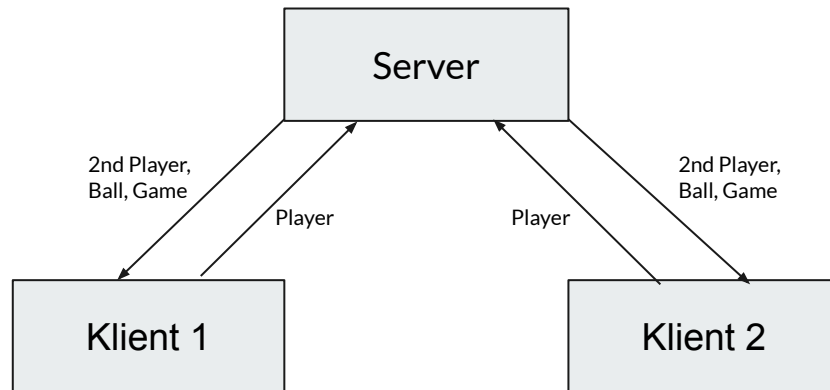


Tryb gry przez LAN w naszej aplikacji

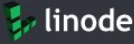


Model wymiany informacji w sieci LAN

- każdy klient wysyła informacje o obsługiwanej przez niego postaci
- dane te są przetwarzane przez serwer i wpływają na stan gry, tj. położenie piłki, wynik
- aktualne dane dotyczące stanu gry są rozsyłane do klientów




Hosting serwera online




Dashboard
Linodes
Volumes
Object Storage
NodeBalancers
Domains
Marketplace
Longview
Kubernetes
StackScripts
Images

Search for Linodes, Volumes, NodeBalancers, Domains, Tags...

Create ▾




0



Dashboard

Linodes

	ubuntu-eu-central Ubuntu 19.10, Nanode 1GB: 1 CPU, 25GB Storage, 1GB RAM	Frankfurt, DE
---	--	---------------

Volumes

No items to display.

NodeBalancers

No items to display.


Domains

No items to display.

Monthly Network Transfer Pool

1 GB Used999 GB Available

Your transfer is prorated and will reset next month

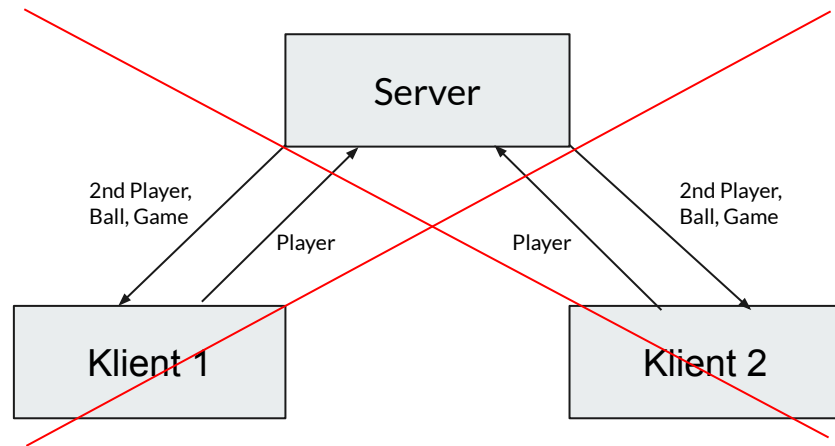
 **Back Up Your Data and Keep it Safe**

Linode Backup Auto-Enrollment

If you enable this global setting, new Linodes will be automatically enrolled in the Backups service.

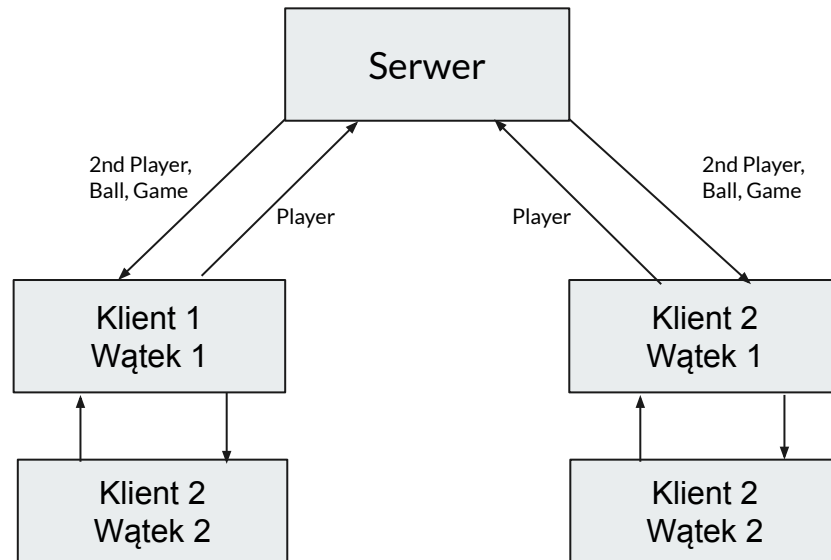
Tryb gry przez Internet

- pojawia się problem związany z pingiem
~20-40ms w przypadku naszego serwera
- opóźnienia powodują, że gra działa < 50 fps
- do płynnej gry potrzeba ~75 fps
- musimy wprowadzić ulepszenia



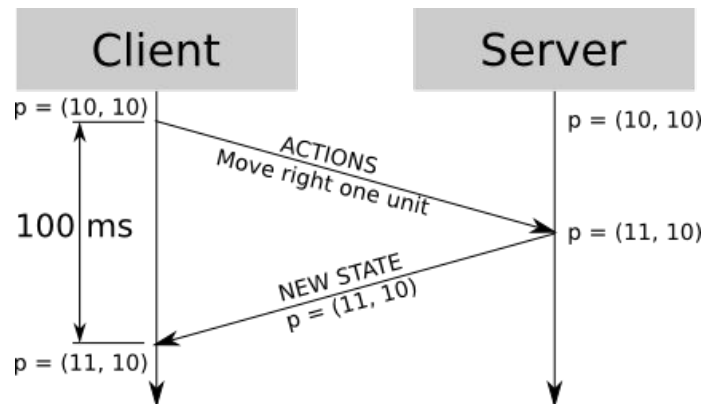
Model wymiany informacji przez Internet

- każdy z klientów generuje ruch swojej postaci
- wygenerowany ruch postaci jest zbierany w paczki i wysyłany serwerowi
- serwer analizuje dane, generuje ruch piłki i wysyła dane do klientów
- wątek pierwszy jest odpowiedzialny tylko za komunikację z serwerem
- wątek drugi może dzięki temu w sposób ciągły zbierać informacje o ruchu gracza



Rozwój gry przez sieć

- kontrola otrzymywanych od klientów danych pod kątem poprawności z zasadami gry
- uniemożliwienie wykonywania niedozwolonych ruchów
- zaimplementowanie techniki client-side prediction





Źródła

- <https://techwithtim.net/tutorials/python-online-game-tutorial/client/>
- <https://realpython.com/python-sockets/#echo-client-and-server>
- <https://www.youtube.com/watch?v=KQaslwElg3w>
- <https://www.youtube.com/watch?v=Jns9LOeslgM>
- <https://www.pygame.org/docs/>
- <https://www.youtube.com/watch?v=i6xMBig-pP4&list=PLzMcbGfZo4-lp3jAExUCewBfMx3UZFkh5>
- <http://openbookproject.net/thinkcs/python/english3e/pygame.html>
- <http://garrettshields.me/blog/2018/03/18/client-side-prediction.html>
- <https://www.gabrielgambetta.com/client-side-prediction-server-reconciliation.html>