Aleksei Haidukevich 295233

31/03/2020

Projekt 1: Wirtualna Kamera

Grafika Komputerowa

Spis Treści:

Opis zadania

Projekt 1 ma na celu implementację tzw. wirtualnej kamery, czyli symulacji prawdziwej kamery, przez którą widz obserwuje pewne obiekty.

Prawdziwa kamera może realizować:

- translacje według każdej z trzech osi,
- rotację wokół każdej z trzech osi,
- zoom (nie jest Z-translacją!)

Czyli wirtualna kamera realizuje te same funkcjonalności.

Program ma zawierać definicje obiektów typu okienko (Window), pisak (Pen), wczytywać dane wejściowe - współrzędne obiektów (najlepiej - z pliku), mapowanie przekształceń na klawisze, przeliczanie zmian i nowych współrzędnych na bieżąco.

Technologia i Implementacja

Jako framework używam modułu **Pygame** napisany w języku **Python**.

Aby uruchomić program, użytkownik musi mieć zainstalowane paczki **pygame** oraz **numpy**. Uruchomienie poleceniem **python Lab1.**

Na samym początku, definiuję wartości, takie jak rozmiar okienka, albo początkowe, takie jak ogniskowa (focal distance):

```
width = 1400
height = 1000
step = 40
rotation_angle = math.pi/36
focal = 300
focal_step = 10
focal_min = 20
focal_max = 470

safe_mode = True
```

Wartości te służą przy definicji obiektu klasy Window, którego używam przy wyświetlaniu.

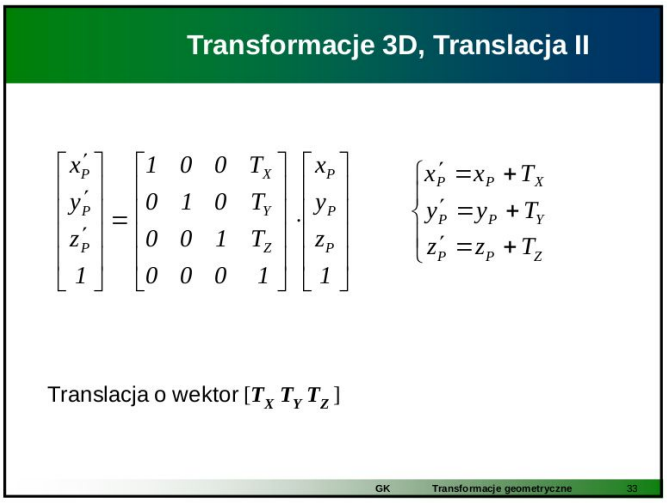
Współrzędne pokazywanych figur wczytuję z jednego z plików, zawierających 1, 2 lub 4 sześciany.

```
-500 100 800
-500 100 1000
-500 -100 800
-500 -100 1000
-300 100 800
-300 100 1000
-300 -100 800
-300 -100 1000
0 1
0 2
0 4
3 2
3 1
3 7
6 2
6 7
6 4
5 7
5 1
5 4
```

Najpierw są podane 8 wierzchołków, a dalej 12 krawędzi. Z tych danych tworzą się macierzy (**numpy.array**), rozszerzone o jedną kolumnę jedynek:

```
<Wireframe object> with nodes at:
[[-500  800  100    1]
 [-500 1000  100    1]
 [-500  800 -100    1]
 [-500 1000 -100    1]
 [-300  800  100    1]
 [-300 1000  100    1]
 [-300  800 -100    1]
 [-300 1000 -100    1]]
```

Wszystkie transformacje (translacje i rotacje) dokonują się na skutek mnożenia macierzy transformacji oraz wektora, który reprezentuje konkretny punkt, np. translacja o wektor [Tx Ty Tz]:



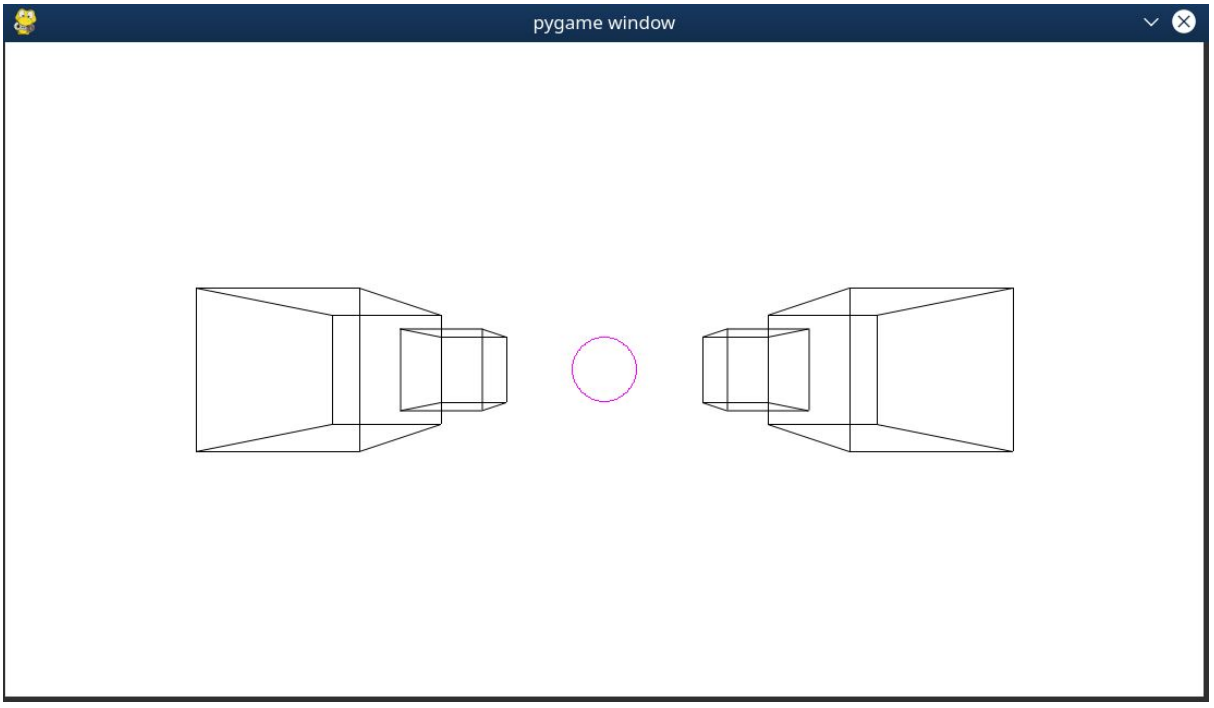
Używając funkcji z modułu **pygame**, program wyświetla okienko, oraz cyklicznie odświeża obrazek, robiąc następujące rzeczy:

- Wypełnia tło kolorem białym [255, 255, 255]
- Rysuje różowy [255, 0, 255] okrąg pośrodku ekranu, promień którego zależy od wartości ogniskowej
- Dla każdego wierzchołka z każdej z figur, sprawdza, czy jest widoczna, i rysuje wierzchołki w rzucie perspektywicznym
- To samo robi dla krawędzi

Pisak (obiekt klasy Pen) używa funkcji z modułu **pygame**:

- **pygame.draw.circle** - rysuje okrąg (wierzchołek, na obrazkach niżej rysowanie wierzchołków jest wyłączone)
- **pygame.draw.aaline** - rysuje prostą (anti-aliased line)

Na skutek tego program po uruchomieniu wygląda następująco:



Demonstracja działania

Transformacje są uzależnione od przycisków klawiatury w następujący sposób:

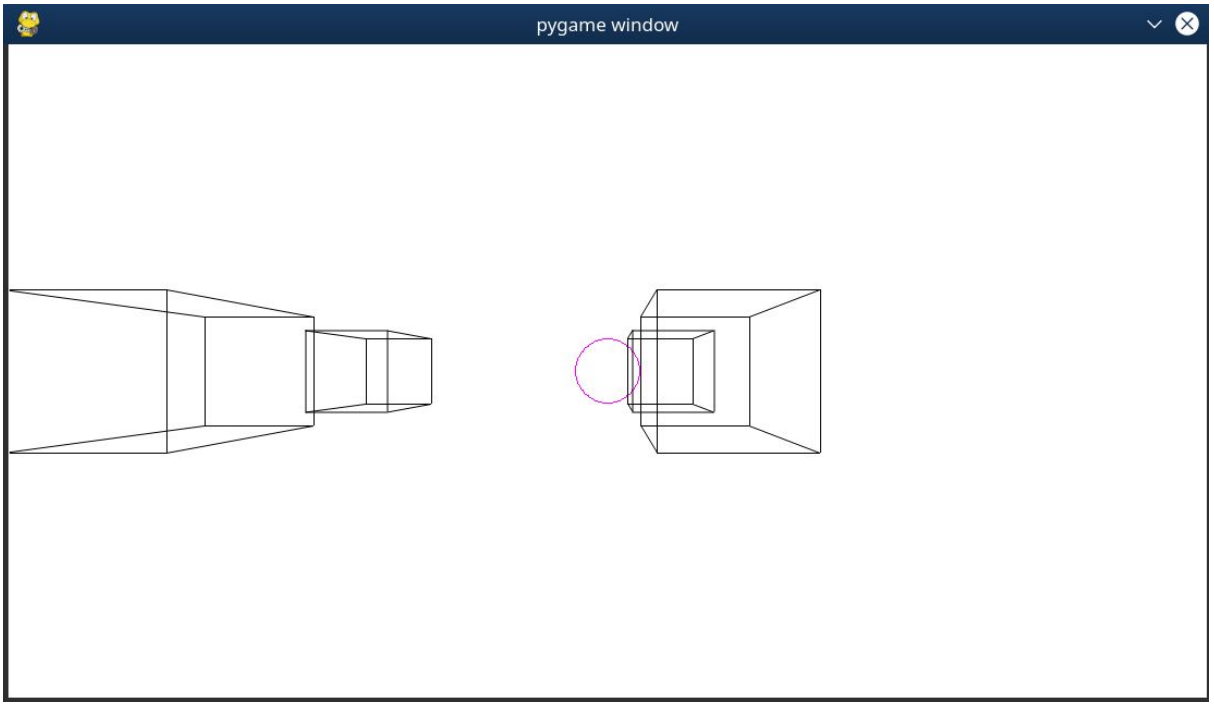
- X-translacja: strzałki **Left** oraz **Right**
- Z-translacja: strzałki **Up** oraz **Down**
- Y-translacja: **Space** oraz **Left Shift**
- Rotacje: klawisze **1 - 6**
- Zoom in: **+ (Plus)**
- Zoom out: **- (Minus)**

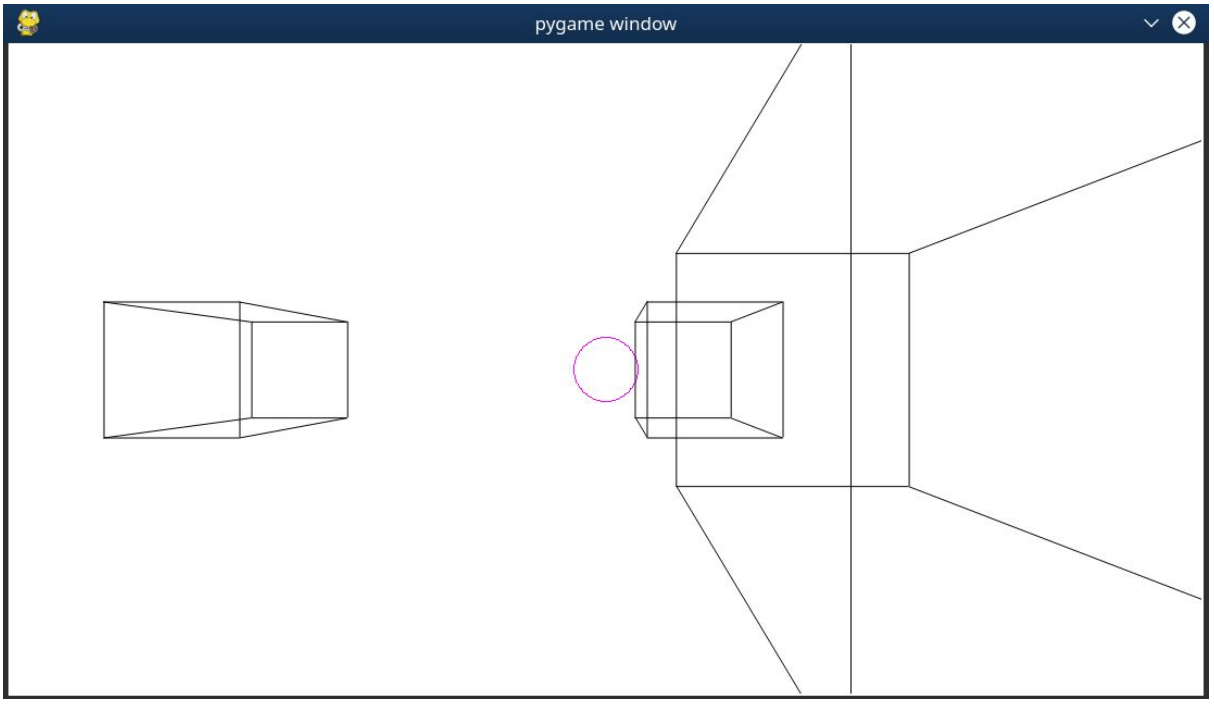
Przy translacji punktu (wierzchołka) odbywa się mnożenie wektora tego punktu oraz macierzy zawierającej wektor translacji.

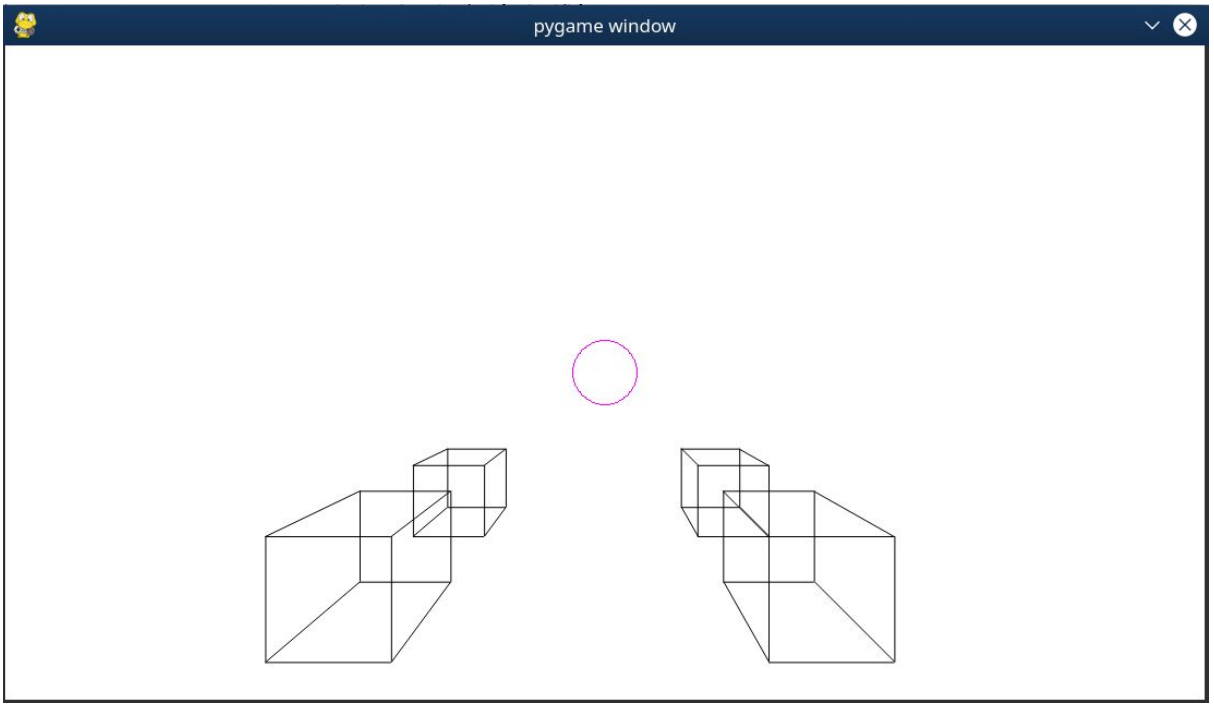
Przy rotacji punktu, macierz transformacji zawiera wartości funkcji trygonometrycznych pewnego kąta rotacji.

Zoom zmienia ogniskową, co powoduje zmiany w rysowaniu punktów przy obliczeniu współrzędnych na rzutni (przy rzutowaniu perspektywicznym).

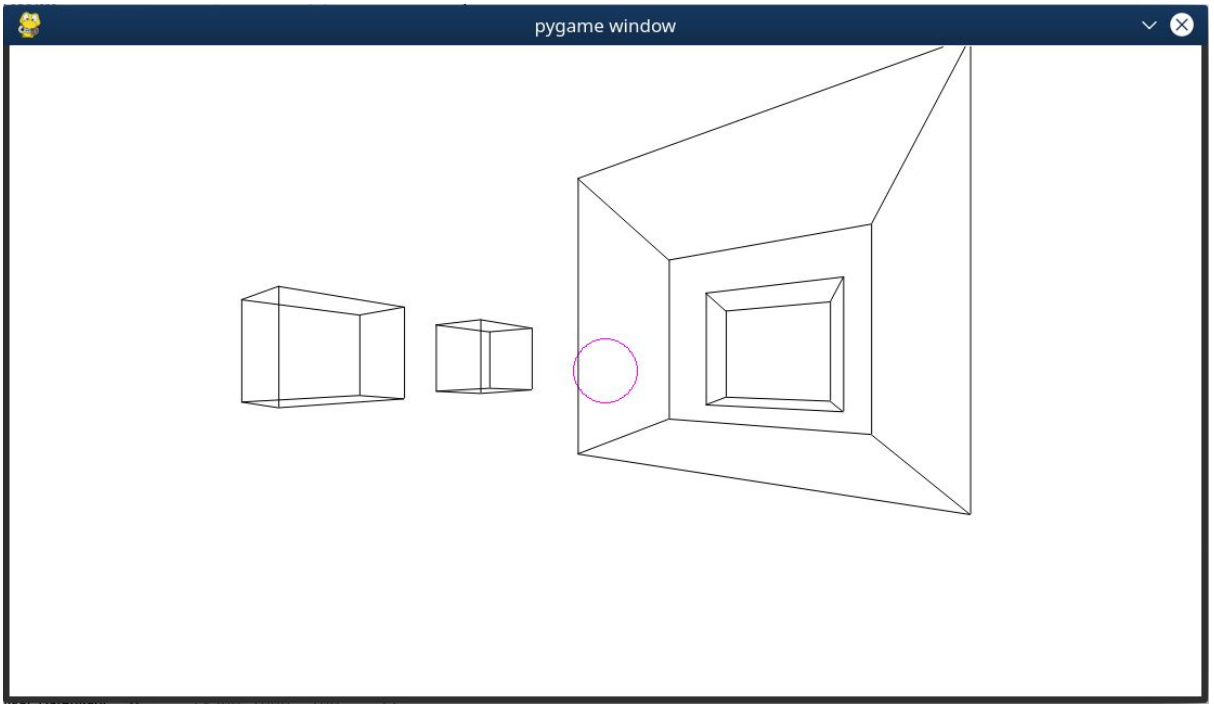
Translacje (bez Safe Mode):

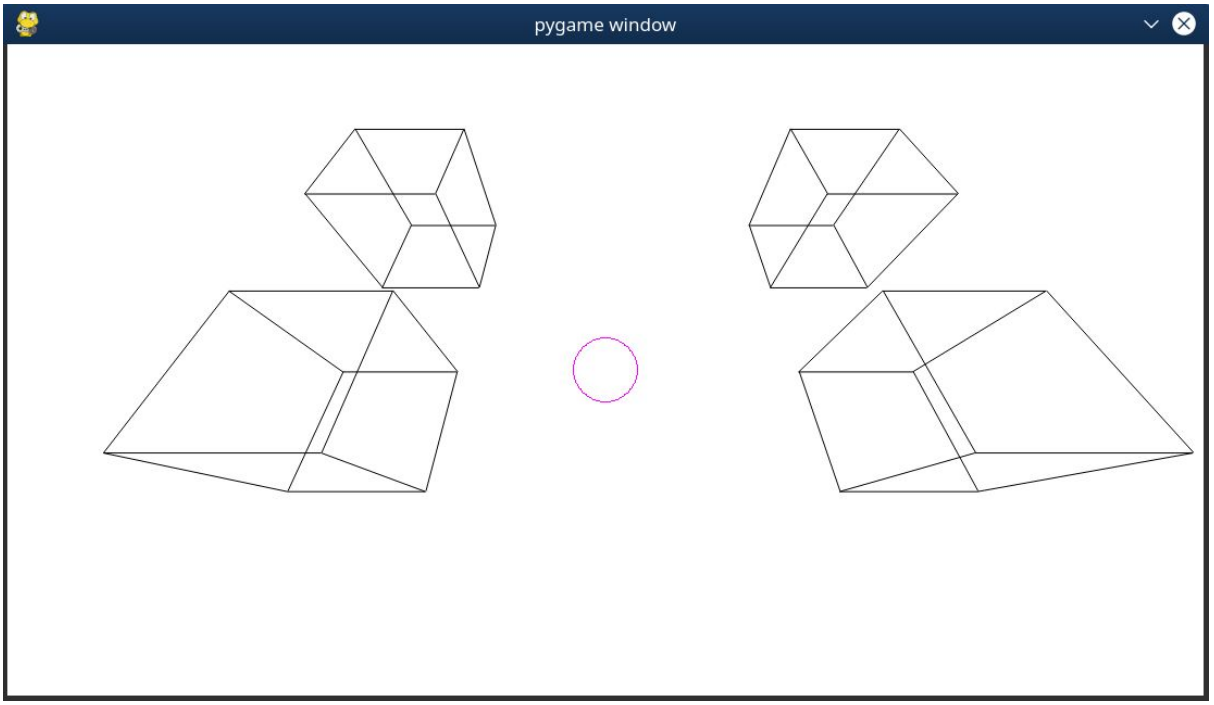


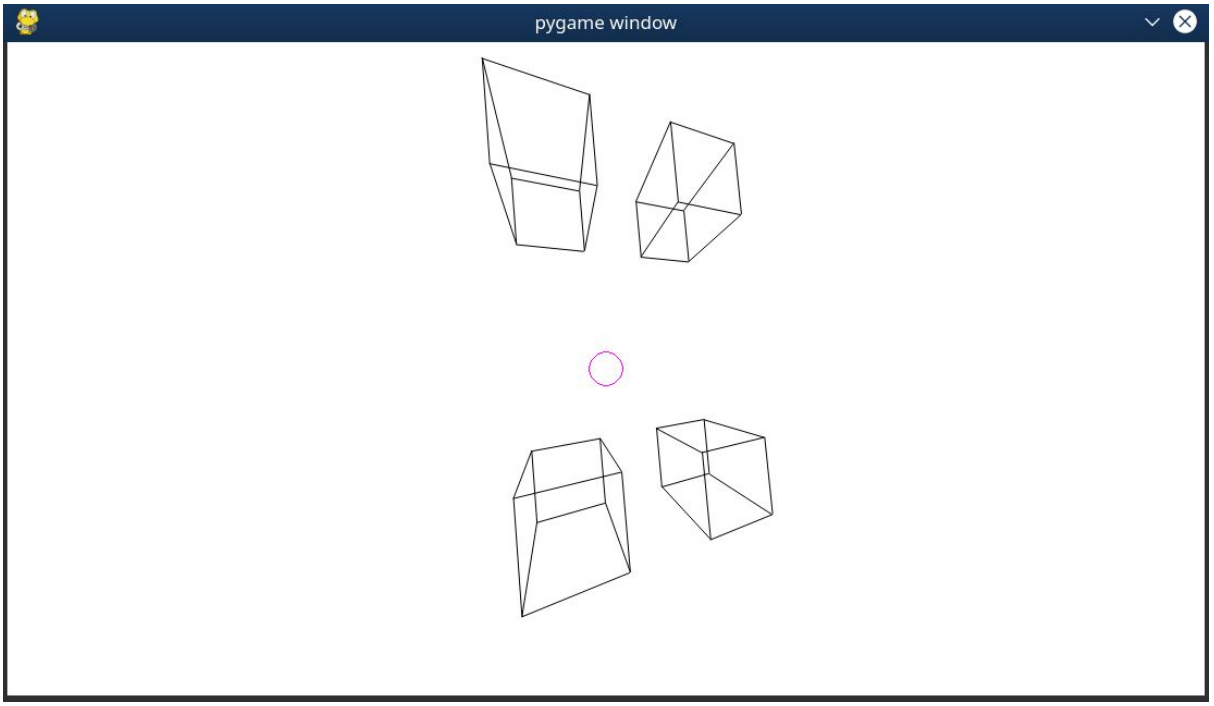




Rotacje:

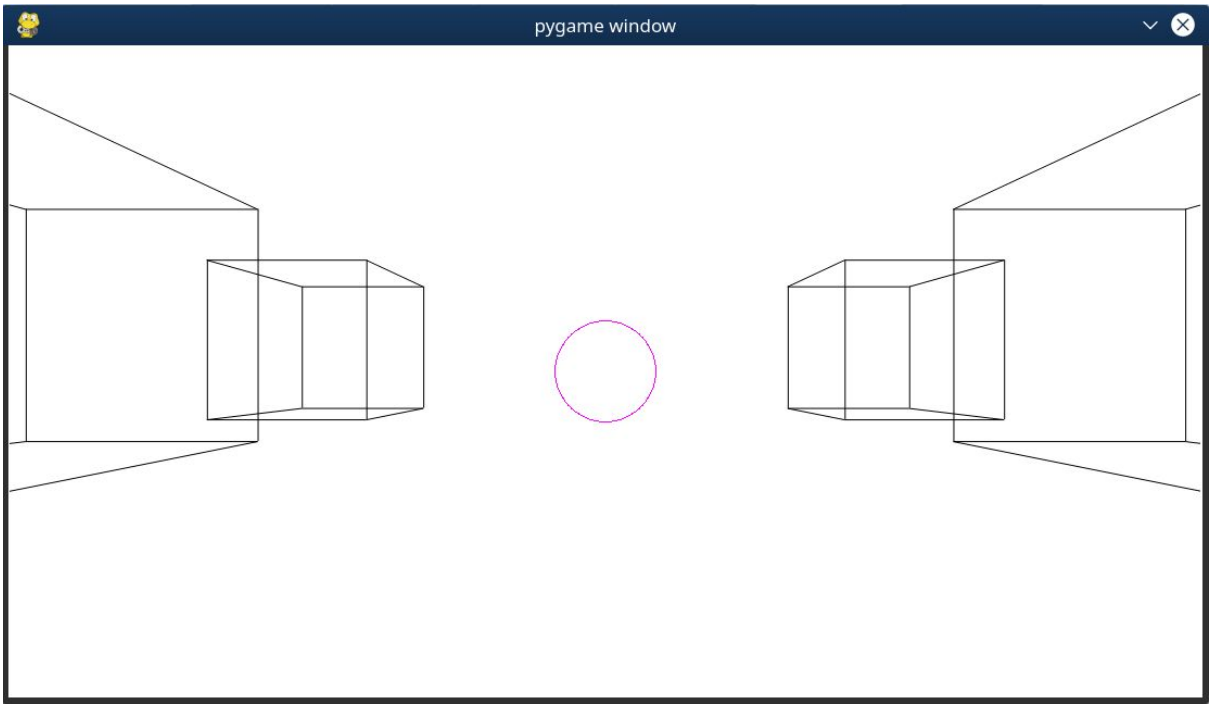


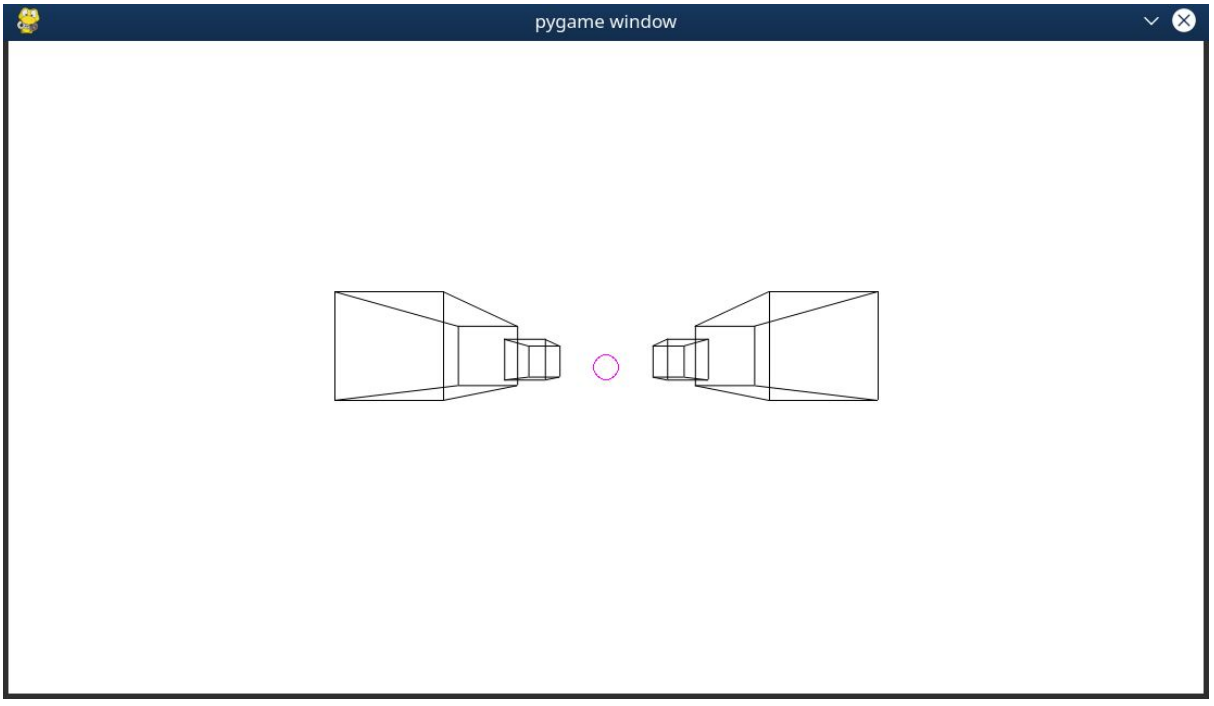

pygame window


pygame window

Zoom:

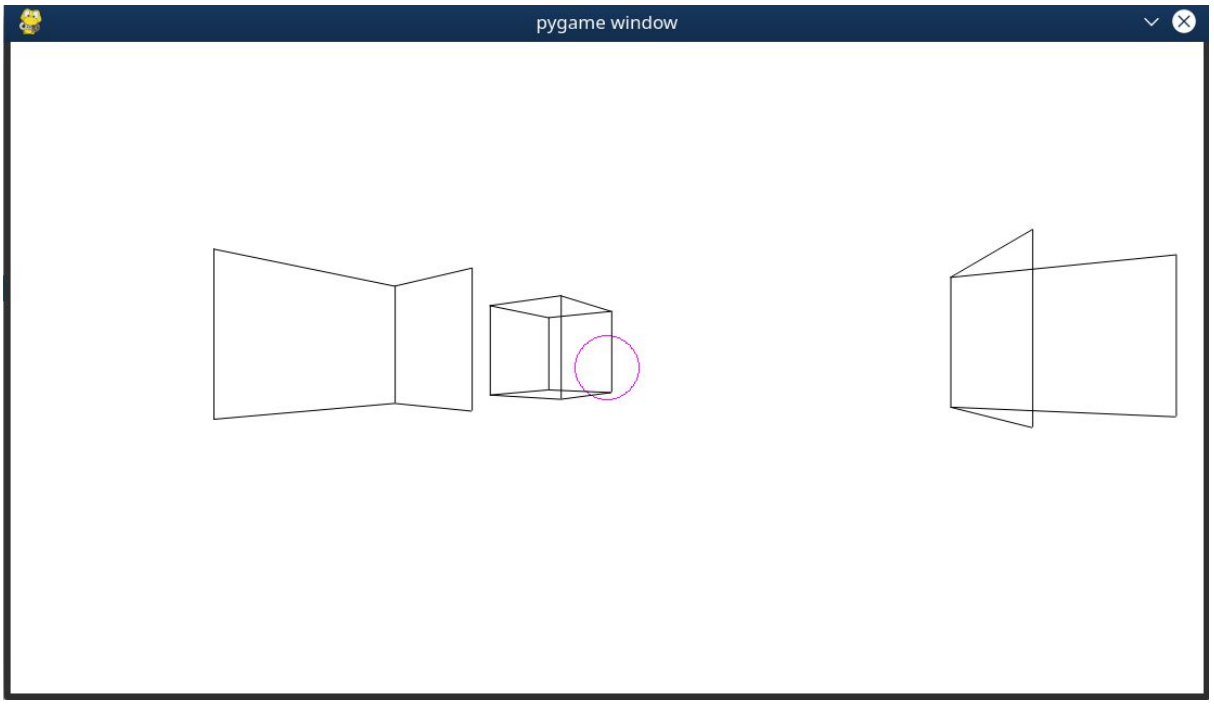
Warto zauważyć, że zoom (w odróżnieniu od Z-translacji) nie pozwala zobaczyć większą część sześcianów stojących dalej, na obrazku jest zasłonięte ~30% powierzchni ściany, i ta proporcja zachowuję się przy użyciu zoomu.





Usuwanie “niewidocznych” wierzchołków w Safe Mode:

Safe mode nie pozwala “kamerze” zbliżyć się do krawędzi, aby nie było nieprawdziwych reprezentacji punktów oraz zepsucia programu:



Przykład niespodziewanego zachowania bez Safe Mode (odbywa się przy rotacji w bezpośrednim pobliżu sześcianu, też jest szansa, że program wyłączy się):