Obiekty wizualne

Circle

Konstruktor

```
Circle(r: float, position: Point = Point(0, 0), color: Color = Color(0, 0, 0, 255))
```

- Parametry
  - `r : float`
    Promień okręgu
  - `position : Point` *opcjonalny*
    Pozycja początkowa koła. Określa położenie punktu znajdującego się w lewym górnym rogu prostokąta opisującego okrąg. Domyślnie przyjmuje wartość `Point(0, 0)` - jest to pozycja w lewym górnym rogu ekranu.
  - `color : Color` *opcjonalny*
    Kolor, jaki zostanie wykorzystany do rysowania koła, Domyślna wartość to `Color(0, 0, 0, 255)`, jest to w pełni nieprzezroczysty kolor czarny.

Zmienne instancji

- `center: Point(x: int, y: int)`
  Obliczana w konstruktorze, określa punkt (w odniesieniu do prostokąta określającego okrąg), w którym znajduje się środek okręgu.

Line

Klasa Line definiuję prostą, przebiegającą przez podane w konstruktorze punkty.

Konstruktor

```
Line(A: Point, B: Point, position: Point = Point(0, 0), color: Color = Color(0, 0, 0, ,255))
```

- Parametry
  - `A : Point`
    Punkt A, przez który przebiega prosta.
  - `B : Point`
    Punkt B, przez który przebiega prosta.
  - `position`
    Pozycja początkowa prostej. Określa przesunięcie prostej od pozycji początkowej Domyślnie przyjmuje wartość `Point(0, 0)` - jest to brak przesunięcia.
  - `color : Color` *opcjonalny*
    Kolor, jaki zostanie wykorzystany do rysowania prostej, Domyślna wartość to `Color(0, 0, 0, 255)`, jest to w pełni nieprzezroczysty kolor czarny.

Line_Segment

Klasa Line_segment definiuje odcinek zaczynający się od punktu A, a kończący się na punkcie B.

Konstruktor

```
Line_Segment(A: Point, B: Point, position: Point = Point(0, 0), color: Color = Color(0, 0, 0, 255)):
```

- Parametry
  - `A : Point`
    Punkt A - początkowy punkt odcinka
  - `B : Point`
    Punkt B - końcowyn punkt odcinka
  - `position` Pozycja początkowa odcinka. Określa przesunięcie odcinka od pozycji początkowej. Domyślnie przyjmuje wartość Point(0, 0) - jest to brak przesunięcia.
  - `color : Color` *opcjonalny*
    Kolor, jaki zostanie wykorzystany do rysowania odcinka, Domyślna wartość to `Color(0, 0, 0, 255)`, jest to w pełni nieprzezroczysty kolor czarny.

Spline

Klasa Spline definiuje krzywą będącą funkcją wielomianu. Przebiega ona przez wszystkie podane w parametrze punkty.

Konstruktor

```
Spline(points: List[Point], tension: float = 0.5, num_segments: int = 100, position: Point = Point(0, 0), color: Color = Color(0, 0, 0, 255))
```

- Parametry
  - `points : List[Point]`
    Lista punktów, przez które będzie przebiegać krzywa
  - `tension : float` *opcjonalny*
    parametr, określający stałą napięcia krzywej. Domyślna wartość, to `0.5`
  - `num_segments : int` *opcjonalny*
    parametr, określający na ile segmentów zostanie podzielona krzywa. Domyślna wartość, to `100`
  - `position` *opcjonalny*
    Pozycja początkowa krzywej. Określa przesunięcie odcinka od pozycji początkowej. Domyślnie przyjmuje wartość Point(0, 0) - jest to brak przesunięcia.
  - `color : Color` *opcjonalny*
    Kolor, jaki zostanie wykorzystany do rysowania krzywej, Domyślna wartość to `Color(0, 0, 0, 255)`, jest to w pełni nieprzezroczysty kolor czarny.

Rect

Klasa Rect definiuje prostokąt, którego lewy górny róg znajduje się w punkcie A, natomiast prawy dolny róg znajduje się w punkcie B. Ściany prostokąta domyślnie są ortogonalne do osi układu współrzędnych.

Kontruktor

```
Rect(A: Point, B: Point, color: Color = Color(0, 0, 0, 255))
```

- Parametry
  - `A: Point`
    Punkt A - punkt określający lewy górny róg prostokąta
  - `B: Point`
    Punkt B - punkt określający prawy dolny róg prostokąta
  - `position : Point` *opcjonalny*
    Pozycja początkowa prostokątu. Określa położenie punktu znajdującego się w lewym górnym rogu prstokątu. Domyślnie przyjmuje wartość `Point(0, 0)` - jest to pozycja w lewym górnym rogu ekranu.
  - `color : Color` *opcjonalny*
    Kolor, jaki zostanie wykorzystany do rysowania prostokąta, Domyślna wartość to `Color(0, 0, 0, 255)`, jest to w pełni nieprzezroczysty kolor czarny.

Triangle

TODO

Klasa Triangle definiuje trójkąt, Trójkąt może zostać z powodzeniem utworzony za pomocą klasy Polygon, dla tego klasa Trójkąt skupia się na szczególnych przypadkach trójkąta, takich, jak trójkąt równoramienny, trójkąt prostokątny etc.

Polygon

Klasa Polygon definiuje wszystkie możliwe wielokąty składające się z n punktów. jego wierzchołki znajdują się w podanych w argumencie punktach. ostatni punkt jest łaczony linią z punkem pierwszym

Konstruktor

```
Polygon(vertices: List[Point], position: Point = Point(0, 0), color: Color = Color(0, 0, 0, 255))
```

- Parametry
  - `vertices : List[Point]`
    Lista punktów, które określają wierzchołki wielokątu
  - `position : Point` *opcjonalny*
    Pozycja początkowa wielokątu. Określa położenie punktu znajdującego się w lewym górnym rogu prostokątu opisującego wielokąt. Domyślnie przyjmuje wartość `Point(0, 0)` - jest to pozycja w lewym górnym rogu ekranu.
  - `color : Color` *opcjonalny*

Kolor, jaki zostanie wykorzystany do rysowania wielokątu, Domyślna wartość to `Color(0, 0, 0, 255)`, jest to w pełni nieprzezroczysty kolor czarny.

Obiekty strukturalne

Grid

Klasa grid definiuje strukturę siatki, pozwala ona na rozmieszczanie obiektów w poszczególnych kolumnach oraz rzędach.

Konstruktor

```
Grid(row_count: int, column_count: int, dimensions: Rect, position: Point = Point(0, 0))
```

- Parametry
  - `row_count: int`
    ilość rzędów siatki
  - `column_count: int`
    ilość kolumn siatki
  - `dimensions: Rect`
    Prostokąt, w którym zawierała będzie się siatka, wymiary komórek skalują się w zależności od tego prostokąta
  - `position : Point` *opcjonalny*
    Pozycja początkowa siatki. Określa położenie punktu znajdującego się w lewym górnym rogu prstokątu w którym zawiera się siatka. Domyślnie przyjmuje wartość `Point(0, 0)` - jest to pozycja w lewym górnym rogu ekranu.

Metody

add(obj, row: int = None, column: int = None) Metoda add dodaje umieszcza wskazany objekt `obj` na pozycji wskazanej przez parametry `row` i `column`. Jeżeli dana pozycja jest już zajęta, to w zależności od parametru `replace` obiekt wczesniej tam znajdujący zostanie usunięty (`relpace = true`), lub dodawanie elementu się nie powiedzie i zostanie podniesiony wyjątek.

- Parametry
  - `obj`
    Objekt, który ma zostać dodany
  - `row: int` *opcjonalny*
    Określa rząd, w którym ma zostać dodany przekazany obiekt. Jeżeli nie została podana żadna wartość, to obiekt zostanie dodany na pierwszej wolnej pozycji w podanej kolumnie.
  - `column: int` *opcjonalny*
    Określa kolumnę, w której ma zostać dodany przekazany obiekt. Jeżeli nie została podana żadna wartość, to obiekt zostanie dodany na pierwszej wolnej pozycji w podanym rzędzie.
  - `replace: bool = true` *opcjonalny*
    Jeżeli ten argument ma wartość `true`, to w sytuacji, gdy dana wskazana pozycja jest już zajęta, poprzednio znajdujący się na niej obiekt zostanie usunięty. Jeżeli ma wartość `false`, to wskazany obiekt nie zostanie dodany i podniesiony zostanie wyjątek. ### remove(row: int = None, column: int = None) Metoda remove usuwa obiekt na wskazanej za pomocą parametrów `row` i `column` pozycji. Jeżeli żaden z tych argumentów nie zostanie podany, to usunięte zostaną wszystkie obiekty w siatce.

- Parametry
  - `row: int` *opcjonalny*
    Określa rząd, w którym zostanie usunięty obiekt. Jeżeli nie została podana żadna wartość, to usunięte zostaną wszystkie obiekty w podanej kolumnie.
  - `column: int` *opcjonalny*
    Określa kolumnę, w której zostanie usunięty obiekt. Jeżeli nie została podana żadna wartość, to usunięte zostaną wszystkie obiekty w podanym rzędzie.

S_Circle

TODO

S_Line

TODO

Obiekty pomocnicze

Animation

Klasa animacja słuzy jako kontener dla obiektów, które będą brały w niej udział. Prztrzymuje również informacje na temat samej animacji.

Konstruktor

```
Animation(frameCount, canvas, bgColor)
```

- Parametry
  - `frameCount: int`
    Określa ilość klatek, z których będzie się składać animacja.
  - `canvas: Canvas`
    Poprzez parametr canvas określane są wymiary tworzonej animacji.
  - `bgColor : Color`
    Kolor, jaki będzie stanowił tło całej animacji.

Metody

```
add(obj, path=None)
```

Metoda add dodaje do animacji obiekt, wraz z obiektem można dodać ścieżkę `Path` jaka będzie dotyczyła danego obiektu.

- Parametry
  - `obj`
    Za pomocą parametru obj przekazuje się obiekt, który zostanie dodany do animacji
  - `path: None` *opcjonalny*
    Za pomocą parametru path przekazywana jest ścieżka `Path`, która będzie określała ruch przekazanego obiektu w czasie, Jeżeli ścieżka nie zostanie określona, to obiekt pozostanie w takiej pozycji, jak jego pozycja początkowa.

getFrames() Metoda pomocnicza, zwraca tablicę wszystkich klatek `Frame` jakie generuje dana animacja.

Path

Reprezentuje ścieżkę, po której będzie się przemieszczał aniowany obiekt.

Konstruktor

```
Path(timePeriod)
```

- Parametry
  - `timePeriod`
    Całkowita ilość klatek `Frame`, na które podzielona zostanie ścieżka (długość życia obiektu)

Metody

```
add(point: Point)
```

Metoda add dodaje do ścieżki kolejny punkt `Point` - poszczególne pozycje w danej ścieżce będą określane pomiędzy tymi punktami.

- Parametry
  - `point: Point`
    Parametr point określa punkt, który będzie brał udział w interpolacji.

asTimeFunction(t: int) Ponieważ biblioteka obsługuje ruch obiektów za pomocą funckji czasu, ta funkcja pozwala na ścieżce `Path` działać, jako taka właśnie funkcja czasu, zwraca ona pozycję w podanym w parametrze `t` czasie.

- Parametry
  - `t: int`
    Za pomocą parametru `t` przekazywany jest moment w czasie, w jakim ma zostać obliczonna pozycja.

Canvas

Konstruktor

Canvas(width: int, height: int)

- Parametry
  - `width: int`
    Określa szerokość obszaru, w jakim prezentowana będzie animacja

- `height: int`
  Określa wysokość obszaru, w jakim prezentowana będzie animacja

Color

TODO

Pixel

Pixel, to obiekt, który określa pozycję oraz kolor znajdujący się na tej pozycji.

Konstruktor

- Parametry
  - `x: int`
    Pozycja horyzontalna (na osi x)
  - `y: int`
    Pozycja wertykalna (na osi y)
  - `color: Color`
    kolor znajdujący się na danej pozycji

Frame

Klasa frame prezentuje daną klatkę animacji

Konstruktor

```
Frame(canvas, bgColor)
```

- Parametry
  - `canvas: Canvas` za pomocą parametru Canvas określany jest obszar, w jakim prezentowana będzie animacja.
  - `bgColor : Color`
    Kolor, jaki będzie stanowił tło całej klatki.

Metody

getImg() Metoda getImg zwraca klatkę jako PiL.Image - klasę z biblioteki pomocniczej PiL.

add(obj, path, t) Metoda add dodaje do klatki obiekt, wraz z obiektem można dodać ścieżkę `Path` jaka będzie dotyczyła danego obiektu. Ponieważ klatka jest określona w konkretnym czasie, to potrzebne jest również przekazanie parametru `t`

- Parametry
  - `obj`
    Za pomocą parametru obj przekazuje się obiekt, który zostanie dodany do klatki
  - `path`
    Za pomocą parametru path przekazywana jest ścieżka `Path`, która zostanie wykorzystana do określenia pzycji obiektu w przekazanym za pomocą `t` momencie w czasie
  - `t: int`
    t określa w jakim czasie mają zostać określone obiekty rysowane w klatce

Serializer

Metody

save(filename: str, animation: animation) Metoda save przetwarza przekazaną animację w gotowy do wyświetlenia plik

- Parametry
  - `filename: str`
    określa nazwę pliku wynikowego animacji.
  - `animation: Animation`
    Za pomocą animation przekazywany jest obiekt animacji, który zostanie zanimowany

Obiekty matematyczne

Point

Point, to obiekt, który określa pozycję

Konstruktor

```
Point(x, y)
```

- Parametry
  - x
    Pozycja horyzontalna (na osi x), przekazany typ jest dowolny, może być to float, lub int
  - y
    Pozycja wertykalna (na osi y)