**ОПИСАНИЕ ТЕХНИЧЕСКОЙ АРХИТЕКТУРЫ**

**class Vector3D - Класс радиус-вектора в 3D-пространстве**

def \_\_init\_\_( x, y, z):

''' создание вектора в трёхмерном пространстве, начало вектора в начале координат '''

Переопределение основных математических операций с векторами

def \_\_add\_\_(self, other) -> Vector3D: # сложение двух векторов

def \_\_sub\_\_(self, other) -> Vector3D: # вычитание одного вектора из другого

def \_\_mul\_\_(self, other) -> float: # умножение двух векторов

def \_\_imul\_\_(self, scalar) -> Vector3D: # умножение двух на число

def \_\_abs\_\_(self) -> float: # модуль вектора

Основные методы

def angle\_between(self, other) -> float: # угол между двумя векторами

def normalize(self): # нормализация длины вектора

def rotate(self, beta, gama): # вычисление вектора для картограммы исследования световой чувствительности глаза. функция возвращает вектор на расстоянии угла beta от точки фиксации взгляда и повернутый на угол gama.

def get\_beta\_gama(self, other\_vec) -> float, float: # операция обратная def rotate

**class BetaGama - объединение полярного радиуса и полярного угла в один объект**

def get\_xy(self): ''' преобразование полярных координат в декартовы'''

**Функции, связанные с двумя предыдущими классами**

def generator\_infinity\_fixation\_vec(base\_vec = Vector3D(0, 1, 0), fps = 30, speed = 1): генератор бесконечной последовательности векторов для точки фиксации в виде знака бесконечности

def get\_optimal\_delta\_beta\_gama(angular\_velocity, fps, beta, delta\_beta, delta\_gama):

''' при заданной угловой скорости angular\_velocity и угловых смещениях beta и gama определяет новые beta и gama так чтобы соблюсти угловую скорость на заданном fps '''

**class BlindPoints - класс точек слепой зоны, т.е. там, где глаз не видит**

def \_\_init\_\_(self) -> None: список классов BetaGama, то есть точек в виде пар beta, gama

Набор методов для работы с класссом как списком

def \_\_len\_\_(self):

def \_\_getitem\_\_(self, key):

def \_\_iter\_\_(self):

def append(self, beta, gama):

def pop (self):

def clear(self):

def area\_polygon(self) -> float: Вычисляет площадь полигона вокруг слепого пятна, а также сохраняет точки самого полигона в self.hull

def get\_max\_beta(self): Выдает максимальное значение beta из списка blind\_points

def get\_min\_beta(self): Выдает минимальное значение beta из списка blind\_points

def get\_max\_gama(self): Выдает максимальное значение gama из списка blind\_points

def get\_min\_gama(self): Выдает минимальное значение gama из списка blind\_points

**class KineticInspection - класс основа для кинетических методов обследования**

Создание полей получения отклика от джойстика

@property

def button\_is\_pressed(self):

@button\_is\_pressed.getter

def button\_is\_pressed(self):

Определение реакции на изменение полярного радиуса точки исследования

@property

def beta(self):

@beta.setter

def beta(self, new\_beta):

def area\_polygon(self): вычисляет текущую площадь слепого пятна, передает данные в VisualField

def wait\_new\_calculation(self): Ожидание разрешения произвести расчёт нового положения векторов. Возвращает True если пришла команда остановить работу процесса

def get\_current\_fps(self) -> int:

вычисляем реальный fps, чтобы узнать насколько нужно сдвигать точку для сохранения постоянной угловой скорости

def get\_avg\_beta\_gama(self): Поиск средних значений beta и gama в слепом пятне после завершения предварительного поиска

def get\_border\_blindspot(self): выдает "границы" слепого пятна по min и max beta и gama

**class BlindSpotParameters - данные для поиска слепого пятна**

def \_\_init\_\_(self, beta\_min = 5, beta\_max = 28, gama\_min = 135, gama\_max = 235,

beta\_scan\_min = 10, beta\_scan\_max = 18, gama\_scan\_min = 170, gama\_scan\_max = 200,

beta\_delta\_scan = 0.75, gama\_delta\_scan = 2.2,

eye\_location = vec.Vector3D(-32, 0, 0),

fixation\_vec = vec.Vector3D(0, 1, 0),

time\_reaction = 0.5,

angle\_speed\_research\_vec = 0.7, angle\_speed\_fixation\_vec = 1/4,

fps = 30,

time\_first\_scan = 90, time\_wait = 1) -> None:

**class AllPoints - Дата класс для аккумуляции всех точек исследования и дальнейшей обработки**

**class BlindSpot(KineticInspection) - Поиск слепого пятна**

def do\_inspection(self): Основной метод определения слепого пятна. Запускается в отдельном потоке, ожидает разрешения произвести расчёт нового положения точек фиксации и исследования. Cобирает и обрабатывает данные исследования по слепому пятну

def search\_first\_blind\_spot(self): Поиск первой точки в слепой зоне. Конкретно ищем некоторое количество точек, в которых обследуемый не видит. Когда получаем некоторое минимальное количество (площадь), то переходим к подробному картографированию

def scan\_up\_down(self): сканировние слепого пятна по принципу вверх-вниз, то есть полярный радиус = const

def scan\_left\_right(self): сканировние по принципу влево-вправо то есть полярный угол = const

def move\_to(self, beta, gama, time\_interval=0, draw\_visual\_field = True) :

Движение точки исследования в заданном направлении (то есть в конечные beta, gama) за определённое время. Необходимо для задержки после выхода из слепого пятна, перемещения на новую позицию.Cкорость определяется @self.angle\_speed\_research\_vec Если значение пришли в нужную точку раньше @time\_interval, то точка застывает до исчерпания time\_interval.

def move\_in\_the\_same\_direction(self, time\_interval, draw\_visual\_field = True):

Продолжить движение точки исследования в том же направлении, что и было в течении @time\_interval. Используеи при выходе из "слепой зоны", чтобы пациент имел временной промежуток для реакции на новое исчезновение точки исследования

def calculate\_new(self):Вычисляем новые значения векторов исследования и фиксации исходя из заданных параметров скорости и направления движения точки исследования

def collect\_statistics(self, new\_data): принимает новые значения @new\_data = class (AllPoints) и производит распределение точек по массивам видимых и невидимых пациентом

с учетом времени реакции

**class TimeReaction(KineticInspection) - обучение пациента реагировать нажатием кнопки джойстика/мыши**

**на появление/исчезновения точки исследования, а также**

**получения ориентировочного времени реакции**

def do\_inspection(self):

1) Обучение пациента реагировать нажатием кнопки джойстика на исчезновение точки обследования и держать кнопку нажатой, пока точка исследования не появиться

2) Получения среднего времени реакции пациента на событие

**class TresholdData - данные для проведения порогового обследования**

**class ThresholdInspection (KineticInspection) - класс для проведения порогового обследования**

def do\_inspection(self):

Проведение порогового исследования

**class VisualField - Класс отрисовки визуального поля и результатов обследования**

def clear(self): стирает все отметки на визуальном поле

def create\_field(self): Создание чистого визуального поля

def draw\_grid(self): рисует сетку полярных градусов и углов

def draw\_point(self, beta, gama, color = (255, 0, 0), size = 1):

Отображает на визуальном поле отдельную точку с заданными полярным координатами

def draw\_blind\_points(self, blind\_points): Отображает найденные слепые точки

def draw\_scan\_blindspot(self, blind\_points, beta, gama, color = (255, 0 ,0), size = 2 ): Отображает процесс сканирования слепого пятна.Отрисовка всех "слепых точек" и точки исследования

def draw\_all\_points(self, all\_points): Отображает все точки исследования. Слепые точки отображаются в другом цвете

def draw\_threshold\_points(self, points): отрисовка всех точек порогового обследования

def draw\_hull(self): рисует заданный контур

**class Ui - Класс интерфейса программы**

**class Pacient - Класс пациент, содержит данные пациента: идентификатор, возраст, обследуемый глаз, результат**

**class Joystick - Класс для обработчика события от манипулятора**

def mouse\_event(self, event): событие нажатия кнопки манипулятора

**class Monitor - Класс монитора для обследования**

def get\_display(self, master\_win): создание экземпляра панели для отображения точек

def display\_show(self): отображение точек на панели монитора

