1. **Правила игры**

Игровое поле представляет собой прямоугольник с центром в точке 0,0, границы которого определены параметрами GameContext.Borders. По углам игрового поля в начале игры создаются акторы игроков, а со временем на полу случайным образом появляются бонусы.

Акторы игроков представляют из себя окружности с радиусом radius и пушкой, стреляющей в направлении Forward. Актор имеют возможность двигаться и стрелять в независимых друг от друга направлениях. При попадании в актор противника ему будет нанесён урон, а стреляющий получит очки. Если противник будет уничтожен, то он отправится на респаун: на 3 секунды его актор будет выведен из игры, после чего случайным образом он появится в одной из начальных точек. В это время программа бота не будет исполнятся.

Дополнительно в процессе игры на поле могут быть случайным образом созданы бонусы, которые могут восполнить здоровье игрока или начислить ему бонусные очки.

Всего игровая сессия занимает 120 секунд и победителем считается игрок, который наберёт наибольшее количество очков.

Правила начисления очков

В игре очки начисляются за следующие действия:

А) Игрок своим выстрелом нанёс урон другому игроку. При этом начисляется количество очков, равное единицам нанесённого урона. Может быть получено из ProjectileInfo.damage (по умолчанию – 10). Если выстрел оказался фатальным, то дополнительно начисляется Gamecontext.FragScore (по умолчанию – 100) очков.

Б) При сборе бонуса игроку будет начислено BonusInfo.bonusScoreValue очков и это количество зависит от типа бонуса

Бонусы

В процессе игры на игровом поле в случайное время и в случайной точке может быть создан один из следующих бонусов:

А) Восстановление 10 единиц здоровья
Б) Восстановление 25 единиц здоровья
В) Восстановление 50 единиц здоровья
Г) Начисление 100 очков

При контакте актора игрока с бонусом игроку начисляется соответствующее количество ресурсов, а бонус исчезнет. При восстановлении здоровья итоговое здоровье актора не может превышать параметр ActorInfo.MaxHealth, однако бонус исчезнет в любом случае

2. Интерфейсы и игровой код

В процессе игры пользователь должен написать программу(бота) для управления Актором в заданном игровом мире. Для управления у актора доступен следующий интерфейс:

```
    public interface IControlledActor
    {
        void RotateTo(Vector2 target);//поворачивать актор в направлении target, пока он не будет смотреть на точку target
        void LookAt(Vector2 target);//то же, что RotateTo
        void RotateLeft(float angle); //повернуть актор налево на угол angle
        void RotateRight(float angle);//повернуть актор направо на угол angle
        float AngleTo(Vector2 target);//возвращает угол, на который актору нужно повернуться, чтобы смотреть на точку target. Если знак отрицательный, то поворот требуется совершить налево, если положительный - направо
        float UnsignedAngleTo(Vector2 target);//то же, что AngleTo, но беззнаковый

        void MoveForward(float force); //назначает ускорение force в направлении обзора актора
        void MoveBackward(float force); //назначает ускорение force в направлении, обратном направлению обзора актора
        void MoveLeft(float force); //назначает ускорение force влево от направления обзора актора
        void MoveRight(float force); // назначает ускорение force вправо отнаправления обзора актора

        void MoveTo(Vector2 target, float force);//назначает ускорение force в направлении точки

        void Shoot();//совершить выстрел в направлени, в котором смотрит актор, если перезарядка завершена


        void Message(string text); //используется для вывода отладочной информации или сообщений другим игрокам

        ActorInfo Info { get; }//возвращает всю доступную информацию об акторе игрока
        float MaxVelocity { get; }//возвращает максимальную скорость актора
        float MaxRotationAngle { get; }//возвращает максимальный угол поворота, который актор может совершить за один такт
    }
```

Чтобы получить информацию о текущем положении актора, нужно обратиться к свойству Info актора. Оно содержит следующую информацию:

```
    public struct ActorInfo
    {
        public Vector2 position;//текущая позиция актора в игровом мире
        public Vector2 velocity;//текущая скорость актора
        public Vector2 forwardVector;//направление, в котором смотрит актор
        public Quaternion rotation;//поворот актора
        public string name;//имя актора
        public float radius;//радиус актора
        public int score;//текущий счёт владельца актора
        public float timeToNextShoot;//время, оставшееся до перезарядки оружия. Если =0, то актор может совершить выстрел
        public int health;//текущее здоровье актора
        public int maxHealth;//максимальное здоровье актора
    }
```

Помимо информации о подконтрольном акторе, игроку доступна информация о состоянии игрового мира, которое определяется интерфейсом IGamecontext. Он содержит информаци о противниках, бонусах, снарядах, границах мира, прошедшем и оставшемся времени.

```
public interface IGameContext
    {
        List<ActorInfo> GetEnemiesInfo();//содержит информацию о акторах других игроков в описанной ранее структуре ActorInfo. Может содержать 0 элементов, если текущий игрок – единственный, существующий в игровом мире на данный момент.
        List<ProjectileInfo> GetProjectilesInfo();//содержит информацию о снарядах, существующих на игровом поле
        List<BonusInfo> GetBonusInfo();//содержит информацию о бонусах, существующих на игровом поле

        float ElapsedTime { get; }//время, прошедшее с начала игры
        float RemainTime { get; }//время, оставшееся до окончания игры
        Borders WorldBorders { get; }//координаты границ мира
        int FragScore { get; }//количество очков, которое игрок получает за убийство другого игрока
    }
```

Информация о снарядах содержится в структуре ProjectileInfo:

```
    public struct ProjectileInfo
    {
        public float radius;//радиус снаряда
        public Vector2 position;//его позиция
        public Vector2 velocity;//скорость
        public string owner;//игрок, который выпустил снаряд
        public float remainLifeTime;//оставшееся время жизни снаряда
        public float damage;//повреждения, которые нанесёт снаряд при столкновении с игроком, отличным от owner. Такое же количество очков получит игрок owner в случае попадания
    }
```

Информация о бонусах содержится в структуре BonusInfo:

```
    public struct BonusInfo
    {
        public Vector2 position;//позицияя бонуса
        public float radius;//радиус бонуса
        public int bonusScoreValue;//количество очков, которое игрок получит за бонус
        public int bonusHealValue;//количество здоровья, которое будет восполнено бонусом
    }
```

Таким образом, задача игрока осуществляется в реализации интерфейса IPlayer.

```
public interface IPlayer
    {
        string Name { get; }//Возвращает имя бота, которое используется в таблице лидеров
        void Tick(IControlledActor actor, IGameContext context);//основной метод управляющей программы. Метод Tick вызывается 1 раз в FixedUpdate при условии, что актор игрока жив. FixedUpdateTime Равен 0.02

    }
```

Игра продолжается 120 секунд и после окончания раунда игрок, набравший наибольшее число очков, объявляется победителем.