**Документация по коду системы выносливости (Stamina) и движения (Mover)**

**Обзор**

Данная система реализует механику выносливости (stamina) и движения персонажа с учетом расхода выносливости при спринте. Код разделен на несколько классов, следующих принципам SOLID и разделения ответственности.

**1. Класс Stamina**

**Назначение**

Управляет логикой выносливости персонажа, включая расход и восстановление, а также взаимодействие с визуальным представлением.

**Поля и свойства**

* \_staminaView (StaminaView) - ссылка на визуальное представление шкалы выносливости
* \_staminaModel (StaminaModel) - модель, содержащая логику работы выносливости
* CanUse (bool) - свойство, проверяющее возможность использования выносливости

**Методы**

* **Constructor(StaminaView staminaView, float baseStamina)**  
  Инициализирует систему выносливости с указанными параметрами.
* **Subscribe()**  
  Подписывает StaminaView на события изменения значения выносливости.
* **Unsubscribe()**  
  Отписывает StaminaView от событий изменения значения выносливости.
* **Use()**  
  Уменьшает выносливость с учетом времени кадра (Time.deltaTime).
* **Restore()**  
  Восстанавливает выносливость с учетом времени кадра (Time.deltaTime).

**2. Класс Mover**

**Назначение**

Управляет перемещением персонажа с учетом состояния выносливости и спринта.

**Поля**

* \_player (Player) - ссылка на персонажа
* \_rigidbody (Rigidbody) - физическое тело персонажа
* \_inputSystem (InputSystem) - система ввода
* \_stamina (Stamina) - система выносливости
* \_baseSpeed (float) - базовая скорость движения
* \_sprintMultiplier (float) - множитель скорости при спринте
* \_isSprinting (bool) - флаг состояния спринта

**Методы**

* **Constructor(...)**  
  Инициализирует систему движения с указанными параметрами.
* **SetSprint(bool isSprinting)**  
  Устанавливает состояние спринта.
* **Move()**  
  Обрабатывает движение персонажа:
  + Применяет множитель скорости при спринте, если есть выносливость
  + Расходует/восстанавливает выносливость в зависимости от состояния
  + Рассчитывает и применяет скорость к Rigidbody

**3. Класс StaminaModel**

**Назначение**

Содержит бизнес-логику работы выносливости.

**Поля**

* \_redPercentage (float) - порог (20%), при котором выносливость считается критически низкой
* \_baseValue (float) - базовое максимальное значение выносливости
* \_currentValue (float) - текущее значение выносливости
* \_isEmpty (bool) - флаг пустой выносливости
* \_currentPercentage (float) - текущий процент выносливости (вычисляемое свойство)

**События**

* ValueChanged (Action<float>) - вызывается при изменении значения выносливости

**Методы**

* **CanUse()**  
  Проверяет, можно ли использовать выносливость:
  + Возвращает false, если выносливость ниже порога и флаг \_isEmpty установлен
  + В противном случае возвращает true
* **Increase(float increaseAmount)**  
  Увеличивает значение выносливости:
  + Проверяет корректность входных данных
  + Не превышает максимальное значение
  + Вызывает событие изменения значения
* **Reduce(float reduceAmount)**  
  Уменьшает значение выносливости:
  + Проверяет корректность входных данных
  + Не опускается ниже 0
  + Устанавливает флаг \_isEmpty при достижении 0
  + Вызывает событие изменения значения

**4. Класс StaminaView**

**Назначение**

Отвечает за визуальное представление шкалы выносливости.

**Поля**

* \_healthBarGradient (Gradient) - градиент для изменения цвета шкалы
* \_staminaFilling (Image) - изображение заполнения шкалы

**Методы**

* **Display(float percentage)**  
  Обновляет визуальное представление шкалы:
  + Устанавливает fillAmount в соответствии с процентом
  + Изменяет цвет согласно градиенту

**Взаимодействие компонентов**

1. **Инициализация**:
   * Создается StaminaModel с базовым значением выносливости
   * Создается Stamina, связывающая модель и представление
   * Создается Mover, получающий ссылку на Stamina
2. **Цикл работы**:
   * Mover вызывает Use()/Restore() в зависимости от состояния спринта
   * StaminaModel обновляет значение и уведомляет StaminaView через событие
   * StaminaView обновляет UI в соответствии с текущим значением

**Особенности реализации**

* Используется событийная модель для обновления UI
* Время кадра (Time.deltaTime) учитывается при изменении значений
* Критический порог выносливости предотвращает злоупотребление спринтом
* Гибкая настройка через параметры конструкторов