Middlewares. Middlewares – это функция, которая вызывается перед обработчиком роута. Они имеют доступ к request и response. По сути они являются такими же, как и в express.

Pipes. Pipe должен реализовывать интерфейс PipeTransform. Pipe преобразует входные данные в желаемый результат. Кроме того, это может сойти за валидацию, так как им же возможно генерировать исключение, если данные неверны. Например:

```
@Post()
@UsePipes(new ValidationPipe(createCatSchema))
async create(@Body() createCatDto: CreateCatDto) {
this.catsService.create(createCatDto);
}
```

NestJS относительно новое решение в области бэкенд-разработки, с большим набором функций для быстрого построения и развертывания корпоративных сервисов, которые отвечают требованиям современных клиентов приложений и придерживаются принципов SOLID и приложения двенадцати факторов. NestJS – это express на стероидах [2].

ЛИТЕРАТУРА

1. Community [Электронный ресурс]. – Режим доступа: https://www.digitalocean.com/community/tutorials/how-to-build-a-blog-with-nest-js-mongodb-and-vue-js. – Дата доступа: 17.04.2021.

2. DocumentationNestJS [Электронный ресурс]. – Режим доступа: https://docs.nestjs.com/. – Дата доступа: 17.04.2021.

УДК 004.9

Студ. Е.А. Шуляк
Науч. рук. доц. О.А. Новосельская
(кафедра информатики и веб-дизайна, БГТУ)

ЛЕВЕЛ ДИЗАЙН В КОМПЬЮТЕРНЫХ ИГРАХ

Левел дизайн – это дисциплина разработки игр, связанная с созданием этапов игры. В отличие от дизайна среды, в котором основное внимание уделяется композиции фона и декораций, левел дизайн включает в себя возможности игрока, игровую механику, создание уровней, локаций, миссий, препятствий и обнаруживаемых элементов, которые создают положительный пользовательский опыт.

Игровые уровни как концепция являются пережитком ранних игр, в которых игроку приходилось переходить с одной стороны уровня на другую, избегая или преодолевая угрозы. Уровни будут служить инструментом для отметки прогресса и постепенного увеличения сложности. Они также были необходимы из-за технологических

ограничений того времени. В начале 1980-х дизайн уровней для 2D-игр, таких как SuperMarioBros или Sonic, в основном предполагал размещение различных препятствий, бонусов и врагов на простой карте. С появлением 3D-игр в 1990-х годах дизайнеры уровней, работающие над такими играми, как Doom и Quake, начали создавать карты, похожие на лабиринты, которые игрок мог исследовать, назначая врагов и предметы на определенные позиции [1].

В последние годы дизайн игровых уровней превратился в гораздо более сложный процесс, требующий навыков архитектуры, искусства, программирования, написания сценариев, психологии и графического дизайна. Хороший дизайн уровней редко бывает произвольным, он целенаправлен и продуман. Каждый элемент дизайна уровня – от его географии и ландшафта до препятствий и конкретной сложности – вносит свой вклад в общий игровой процесс [2].

В большинстве случаев основным методом взаимодействия игрока с вашим уровнем будет навигация – процесс фактического прохождения уровня. Для плавного и приятного игрового процесса игрок всегда должен точно знать, куда ему идти. Тщательная планировка, освещение, вывески и другие визуальные подсказки должны создать естественный «поток» на уровне, который инстинктивно направляет игрока через него. С эстетической точки зрения, все уровни игры должны работать вместе, чтобы создать согласованный визуальный язык за счет использования цвета и формы, который игрок может изучить, чтобы интуитивно продвигаться по уровню. Хотя базовое продвижение по уровню должно быть легким, игровой процесс с навигацией также может быть использован для развлечения. Вполне уместно скрыть некоторые области от игрока, добавить глубины и возможности повторного прохождения через исследование (при условии добавления необходимых визуальных или повествовательных подсказок) или создать области, в которых игрок чувствует себя потерянным или сбитым с толку, чтобы создать ощущение драматичности.

Следует использовать неявное и неожиданно возникающее повествование, а не явное. Это можно назвать парадигмой «Разорванный круг», где используются три ключевых аспекта повествования, присутствующие в дизайне уровней:

– явный, где информация предоставляется игроку через ролики, текст, разговорный диалог, например, цель миссии или кат-сцена;

– неявный, посредством которого игрок получает информацию, глядя на окружающую среду через мизансцену;

– emergent (неожиданно возникающее повествование), где история пишется игроком по мере прохождения уровня.

В то время как дизайнер уровней должен позаботиться о создании подробного повествования, поскольку именно он формирует наш «Разорванный круг», именно последние два элемента создают важнейший «разрыв» и действительно выделяют уровень. Использование мизансцены интегрирует историю в игровой мир и стимулирует воображение игрока неявным повествованием. В то время, как в неожиданно возникающем, история пишется игроком посредством выбора игрового процесса: какое оружие использовать, какой маршрут выбрать, в каком стиле решить проблему и т. д. Эти элементы позволяют игрокам заполнить «пробел» своими собственными действиями и воображением.

Исходя из вышеперечисленного, можно сделать вывод, что желаемая эмоциональная реакция игрока на уровень настолько важна, что всегда должна быть отправной точкой дизайна. Основываясь на этом, можно выбрать, какие пространственные метрики, элементы повествования и игровые механики могут быть развернуты для наилучшего создания ответной реакции игрока.

ЛИТЕРАТУРА

1. Nuclino. VideoGameLevelDesign [Электронный ресурс]. – Режим доступа: https://www.nuclino.com/articles/level-design – Дата доступа: 17.03.2021.

2. MasterClass. How to Become a Video Game Level Designer [Электронныйресурс]. – Режим доступа: https://www.masterclass.com/articles/how-to-become-a-video-game-level-designer#how-to-become-a-video-game-level-designer – Датадоступа: 01.04.2021.

УДК 004.5

Студ. Е.С. Храмова
Науч. рук. ст. преп. Н.И. Потапенко
(кафедра информатики и веб-дизайна, БГТУ)

ТЕХНОЛОГИИ ТРЕХМЕРНОЙ ВИЗУАЛИЗАЦИИ В ВЕБ-ДИЗАЙНЕ

WebGL (Web-based Graphics Library) – кроссплатформенный API для 3D-графики в браузере, разрабатываемый некоммерческой организацией KhronosGroup. WebGL использует язык программирования шейдеров GLSL. WebGL исполняется как элемент HTML5 и поэтому является полноценной частью объектной модели документа (DOM API) браузера[1]. Web GL выполняется на графическом процессоре компьютера. То есть нужно написать код, который выполняется на этом процессоре. Код представлен в виде пар функций. Эти две