Создать страницу, содержащую два контейнера: контейнер с блоками-линиями и контейнер с кнопками (Рис.1).

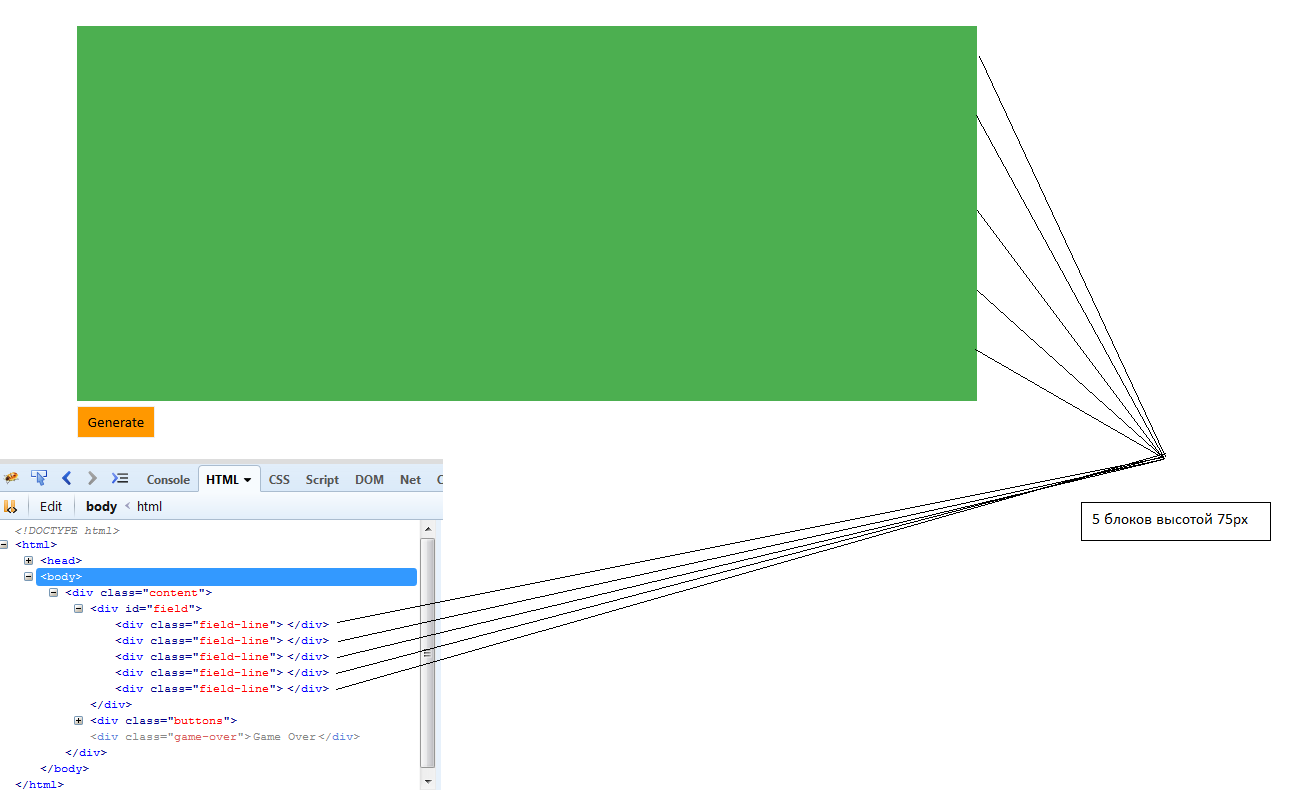


Рис.1

При каждом нажатии на кнопку <Generate> создается случайный зомби в конце случайной линии и начинает движение к началу. Рекомендуемая скорость: 1px каждые 100мсек. (Рис.2, Рис.3).



Рис.2

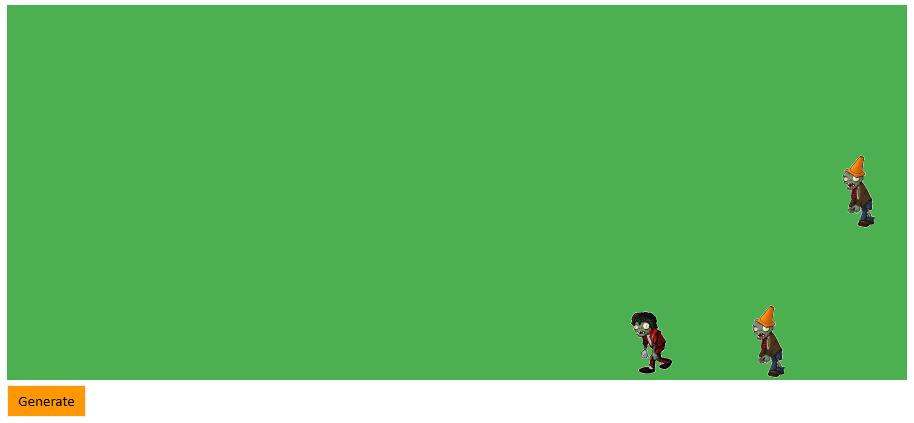


Рис.3

После того, как любой из зомби дошел до начала линии, игра прекращается (Рис.4).

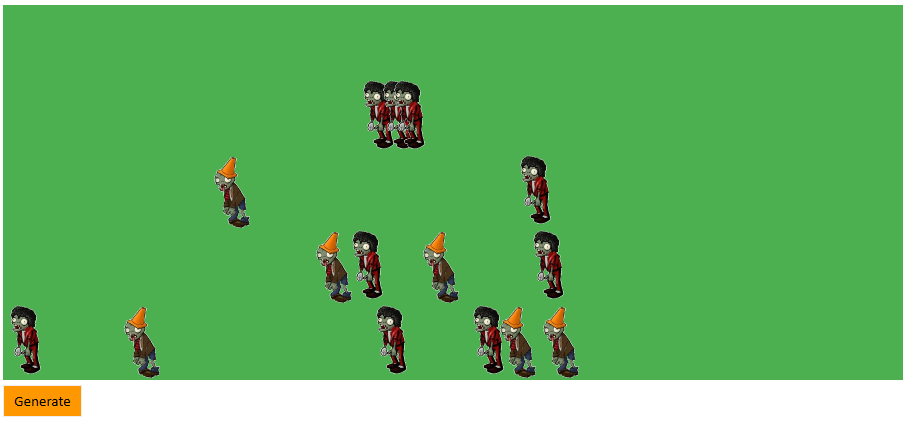


Рис.4

В этом случае поле очищается и выдается сообщение об окончании игры (Рис.5).



Рис.5

***Пояснение к заданию:***

Цель задания – изучить возможность использования объектного подхода при использовании JS.

Итак, существует несколько различных видов зомби, но они подчинены общим для зомби правилам.

Например:

1. каждый вид умеет ходить
2. каждый вид обладает определенным количеством здоровья (в 4 задаче это не требуется)
3. каждый вид выглядит по-разному, но все еще выглядит как зомби
4. каждый вид может умереть (умирать они могут по-разному)

При этом каждый из них не знает о существовании других (даже на своей линии).

Все что знает зомби – линия, по которой он идет и конечная точка.

Это значит, что можно описать общий (абстрактный) класс, который будет описывать общее поведение и свойства зомби.

И каждый из видов зомби должен унаследовать этот функционал.

Поскольку зомби не в курсе о динамике игры и ее настройках, то он может лишь выполнять какие-либо действия.

Например, у каждого зомби должен быть метод “move”. И если его вызвать – зомби переместится на некоторое расстояние.

Или метод “die”, который приведет к тому, что зомби (возможно совершив какие-либо действия) удалит себя с линии.

Для описания подобной игры хорошо подходит паттерн “медиатор”.

Общий принцип такой: отдельный скрипт выполняет роль “движка”. Т.е. принимает решение о необходимости выполнения определенных действий объектами игры (в данном случае - зомби).

Интервал выполнения этих действий, обработчик кнопок и прочее сосредоточено в рамках этого скрипта.

“Движок” знает о всех объектах игры и принимает решение о порядке и последовательности действий ими.

Рекомендую каждую функцию-класс выносить в отдельный JS-файл.

Например, “unit.zombie.michael = function() {…}” можно вынести в файл “unit.zombie.michael.js”.