**ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ**

1. Птица не должна улетать за границы поля. Если птица касается земли, игра заканчивается, если «потолка» — игра продолжается.
2. Если птица касается трубы, игра заканчивается.
3. При клике на окно игры птица подлетает вверх на высоту, равную половине высоты свободного промежутка в трубе.
4. При бездействии игрока птица падает с ускорением.
5. Свободный промежуток в трубе занимает 25 % высоты трубы.
6. Высота птицы составляет 20 % от высоты свободного промежутка в трубе.
7. Ширина трубы вдвое больше ширины птицы.
8. Расстояние между трубами равно ширине трёх труб.
9. Птица двигается с такой скоростью, что новые трубы появляются каждую секунду.
10. Подсчитывается текущее количество баллов. Оно увеличивается, когда птица преодолевает середину свободного промежутка в трубе.
11. Лучший результат игрока (максимальное количество набранных баллов) сохраняется в localStorage и отображается под текущим количеством баллов, если игра запускается не в первый раз.
12. Птица анимирована (вращается при движении в сторону полёта).

**ТРЕБОВАНИЯ К ИНТЕРФЕЙСУ**

Строгих требований к интерфейсу нет, но он должен содержать:

* окно с игрой:



* окно с текущими баллами и лучшим результатом, если игра запускается не впервые:



* кнопку запуска или перезапуска после завершения игры:



**ТРЕБОВАНИЯ К КОДУ**

* 1. Используйте классический *JavaScript* без дополнительных библиотек.
  2. Давайте переменным, классам и функциям осмысленные имена.
  3. Правильно используйте отступы.
  4. Применяйте ООП на ES6-классах.
  5. Следуйте принципам *DRY (Don’t Repeat Yourself)* и *KISS (Keep It Short and Simple)*.
  6. Сделайте обработку получения очков и логику игры независимой от используемого метода отображения игры и используемой в игре физики:
     + - * Вынесите в отдельный класс логику расчёта отрисовки на канвасе, чтобы можно было, например, заменить отрисовку на канвасе на отрисовку в DOM без изменения кода самой игры, просто поменяв класс, отвечающий за отрисовку.
         * Вынесите в отдельный класс логику падения птицы, чтобы, если понадобится более сложная и реалистичная логика расчёта механики птицы, можно было заменить класс, отвечающий за эту логику, и не переписывать остальной код (например, если в будущем мы захотим использовать какой-нибудь физический движок).
  7. Вынесите константы в отдельный файл конфига, разбейте их на логические блоки.
  8. Грамотно разбейте проект на файлы, продумайте и реализуйте их структуру.

**ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ**

* добавить в игру увеличение уровня сложности при достижении определённого количества очков (например, при достижении порога очков в 10, 100, 1000, 1000 и так далее увеличивать скорость движения карты) и альтернативный способ управления, например поддержку геймпада или и клавиатуры, и мыши;
* адаптировать игру под мобильные устройства (от 360 px до 1024 px);
* добавить в игру звуки с использованием *Audio API*.