Bomberman 

>> Настройка окружения.

1. Скачать темплейт проекта:
    a. https://github.com/kyaune/p5play-scheme
    b. Создать `main.js` в папке `/js` и подключить его в `index.html`
2. Установить в Atom:
    a. сниппеты для `p5js` и `p5play`
    b. автоформатирование `atom-beautify`. В настройках доскроллить до JavaScript и поставить галочку – `Beautify on Save`
3. Проверить работоспособность `p5js-toolbar` в Atom
    a. Если не установлен тулбар, то установить как расширения в Atom
    b. Должен запускаться локальный сервер
    c. Открыть в Chrome — `http://localhost:8000/index.html`
    d. Протестировать работоспособность через загрузку изображения из папки со спрайтами

Важно !!
Когда вы будете работать над проектом и после сохранения .js файла, решите проверить результат в браузере. То возможно, что после обновления страницы ничего не изменится. Чтобы изменения отобразились используйте комбинацию клавиш `ctr + shift + r` или `shift + F5`.

Это связано с тем, что браузер кэширует старую версию и показывает её вам. Обновление страницы + shift – команда браузеру запросить с сервера весь сайт целиком.

>> Создание сцены (игровое поле для bomberman)

1. В файле `main.js` выделить комметариями блок для глобальных переменных – так будет удобнее смотреть на код.

2. Объявить глобальные переменные:
    a. для хранения файлов изображений спрайтов:
    ```
    let stoneImg, grassImg, brickImg;
    ```
    b. Имена для групп спрайтов:
    ```
    let greenField;
    let wall;
    let bricks;
    ```
    c. Количество строк, столбцов игрового поля и ширина одного блока:
    ```
    let rows, cols, w;
    ```

3. Написать функцию для предзагрузки изображений, разместить между блоком с объявление глобальных переменных и сетапом:

```
function preload() {
      stoneImg = loadImage(“sprites/Blocks/SolidBlock.png”);
      ..
}
```

4. Создать игровое поле и поместить его в div, у которого id = “game”. Строку с этим div вы можете найти в index.html.

```
..
function setup() {
      let canvas = createCanvas(680, 520);
      canvas.parent('game');
      ..
}
```

5. В сетапе определить значение `w` (ширина одного блока) и создать группы для спрайтов:

```
..
w = width / cols;
greenField = new Group();
wall = new Group();
bricks = new Group();
..
```

6. Написать функцию создания игрового поля – `createScene()`:

```
..
function setup() {
    ..
  createScene();
}

function draw() {
  ..
  drawSprites();
}

function createScene() {
..
}
```

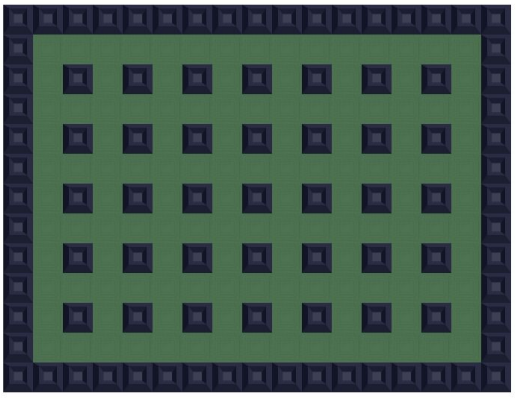

рис. 1

Как сделать игровое поле?

1. У вас есть количество строк `rows` и столбцов `cols` – используйте их во вложенном цикле.
2. В цикле вы создаёте спрайт и определяете через индексы кем ему быть – камнем или травой. Вспомните шахматы :)
3. Добавить в спрайт изображение, изменить размер изображения, затем добавить его в соответствующую группу спрайтов.

Шпаргалка

Методы и аттрибуты, которые понадобятся для функции `createScene()`.

```
// Метод. Создать спрайт
element = createSprite(x, y, width, height);

// Метод. Добавить картинку из переменной stoneImg в спрайт
element.addImage(stoneImg);

// Аттрибут. Изменить масштаб картинки в 2 раза
element.scale = 2;

// Метод. Получить размер изображения в спрайте
let widthSize = element.width;

// Метод. Добавить спрайт element в группу спрайтов wall
wall.add(element);
```

Шпаргалка для самых ленивых.

Код функции `createScene()`.
Но даже в этой шпаргалка надо подумать — какими должны быть индексы, чтобы игровое поле соответствовало `рис.1`?

```
function createScene() {
  let x = w / 2;
  let y = w / 2;

  for (let i = 0; i < rows; i++) {
    for (let j = 0; j < cols; j++) {
      let element = createSprite(x, y, w, w);
      if (значение индекса соответствует позиции для камня) {
        element.addImage(stoneImg);
        element.scale = w / element.width;
        wall.add(element);
      } else {
        element.addImage(grassImg);
        element.scale = w / element.width;
        greenField.add(element);
      }
      x += w;
    }
    x = w / 2;
    y += w;
  }
}
```

>> Создание бомбермена

1. В область глобальных переменных объявить бомбермена и добавить структуру для хранения изображений:
```
let bomberman;
let bombermanImg = {
  back: 0,
  front: 0,
  left: 0,
  right: 0,
};
```

2. Добавить в preload загрузку изображений для анимации:
```
bombermanImg.back =
loadAnimation("адрес/имяПервогоКадра.png",
"адрес/имяПоследнегоКадра.png");
```

3. Добавить в `setup` создание спрайта для бомбермена и добавить анимацию в спрайт:
```
bomberman = createSprite(x, y, width, height);
bomberman.addAnimation(“left”, bombermanImg.left);
```

4. Отмасштабировать бомбермена в `setup` через аттрибут `.scale`

5. Написать функцию для управления бомберменом с помощью клавиш — `bombermanWalkFunction()`. В зависимости от направления меняется анимация.

6. Сделать collide бомбермена со группой стена:
```
bomberman.collide(wall);
```

7. Задать в `setup()` “слои”, в которых будут отображаться спрайты. Спрайт с большим значение `.depth` будет отрисовываться поверх спрайта с меньшим значением:
```
bomberman.depth = 2;
 for (element of greenField) {
    element.depth = 1;
}
```

Как сделать управление и менять анимацию?

1. В зависимости от нажатия клавиши выставлять скорость по x или y
2. В зависимости от значения скорости по x и y выбирать анимацию

Шпаргалка

Методы, аттрибуты и функции, которые понадобятся для функции `bombermanWalkFunction()`.

```
// Функция keyDown(" ") для считывания клавиши
if (keyDowm("w") || keyDowm(UP_ARROW)) {
      .. спрайт идёт вверх
}


// Метод. Установить анимацию с лейблом front
// для спрайта спрайта bomberman
bomberman.changeAnimation("front");

// Методы для запуска и остановки анимации
bomberman.animation.play();
bomberman.animation.stop();

// Аттрибут для задания скорости анимации
bomberman.animation.frameDelay = 2;
```