BMP to sprite converter manual

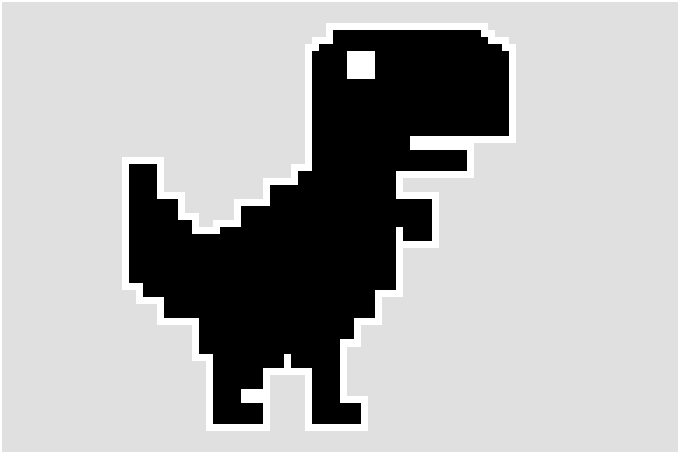
Автор: Левицкий Илья

Вступление

- Здесь описано как пользоваться преобразованием из картинки в спрайт.
- *Спрайт* – ~~напиток~~ графический объект в компьютерной графике.
- Преобразование = 2 компонента
  - **Программа bmp2arr** переводит BMP в C файл.
  - Этот C файл и **новые библиотеки oled_display** позволяют отобразить на экране ваши рисунки

Быстрый старт

- Создайте картинку в формате BMP
- Глубина цвета - 8 бит.
- Имя картинки = имя спрайта.
- Три «цвета»: белый, черный и прозрачный. (Прозрачный – любой цвет кроме черного и белого)



dino.bmp

Быстрый старт

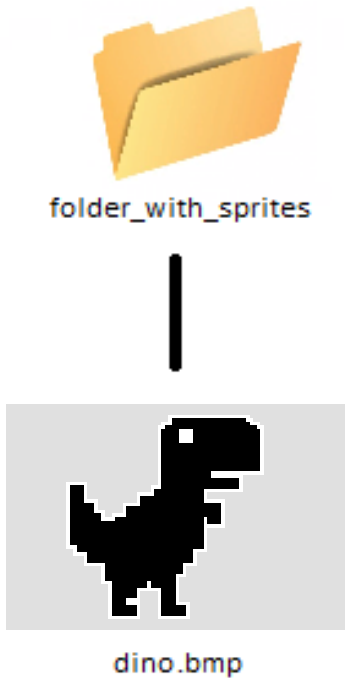
- Сохраните картинку в отдельной папке, в которой больше нет других файлов.

Быстрый старт

Запустите

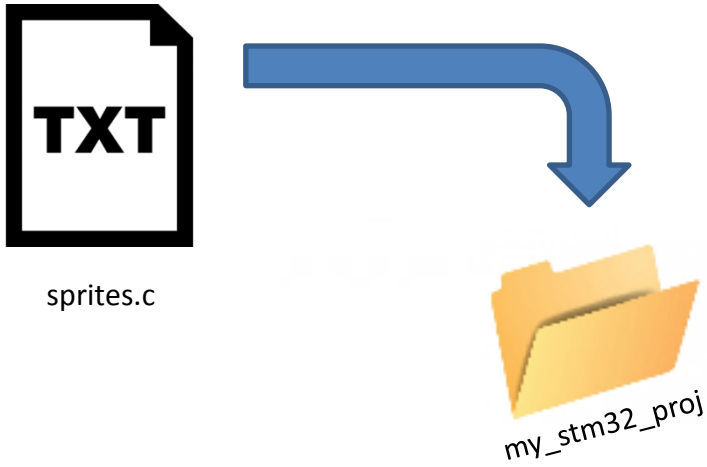
```
$> ./bmp2arr folder_with_sprites sprites.c
```

(программа) (имя корневой папки) (имя выходного файла) [необязателльно]



sprites.c

Быстрый старт



- Переместите С файл в Ваш МК проект (если он еще не там)

- Используйте oled_driver с поддержкой рисования спрайтов

oled_driver.h × oled_driver.c ×

```
/*
 * Draw sprite at specified coordinates
 * with proper costume and its color
 */
void draw_sprite(Sprite sprite, uint32_t costume, int16_t x, int16_t
y, uint32_t options_mask);

/*
 * Test if sprite costume collides with
 * already drawn pixels in picture
 */
uint32_t check_sprite_collision(Sprite sprite, uint32_t costume,
int16_t x, int16_t y, uint32_t options_mask);

#endif
```

Быстрый старт

- Приступим к коду!

```
/* This is an approximate code  */
#include “oled_display.h”

<…>

/* Export init function from generated file */
extern void init_dino_sprite(Sprite *sprite);

Sprite dino;
int main() {

  <…>

  oled_config();
  init_dino_sprite(&dino);

  int16_t x = 10; int16_t y = 10;
```

Не забудьте импортировать функции, плз

Новый тип для спрайта!

Создадим спрайт

Проинициализируем спрайт именно нашей картинкой

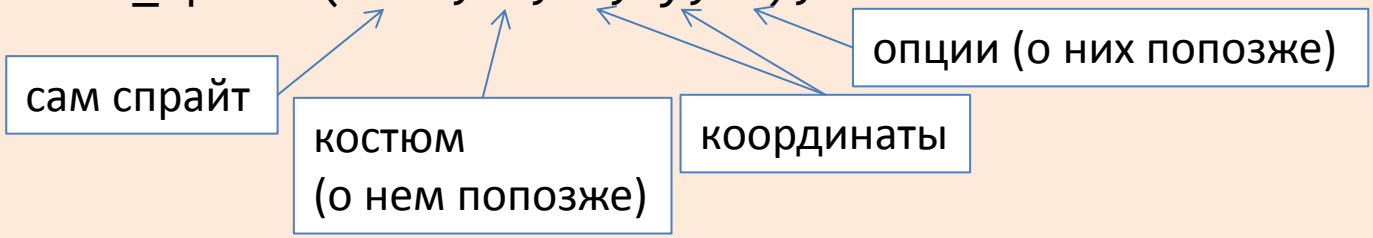
Быстрый старт

```
  init_dino_sprite(&dino);

  /* Coordinates of the sprite */
  int16_t x = 10; int16_t y = 10;

  draw_sprite(dino, 0, x, y, 0);

  oled_update();
  <…>
}
```



Быстрый старт

- Компилируем,
- Запускаем,
- Любуемся!!

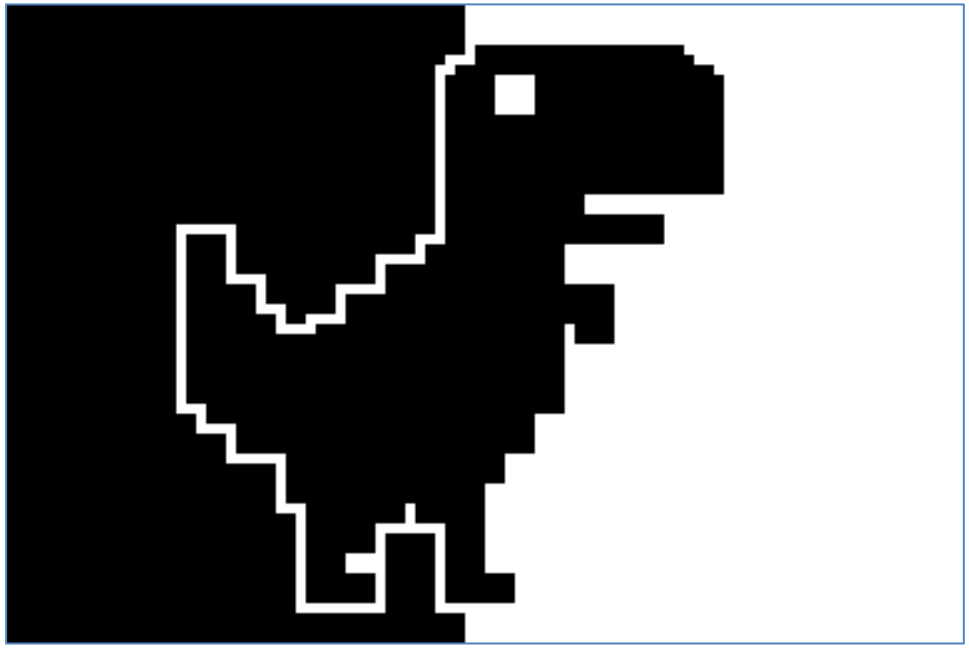
Опции

- Позволяют менять цвет спрайта и отражать спрайт
- Будем рисовать на этом фоне…

… этот спрайт

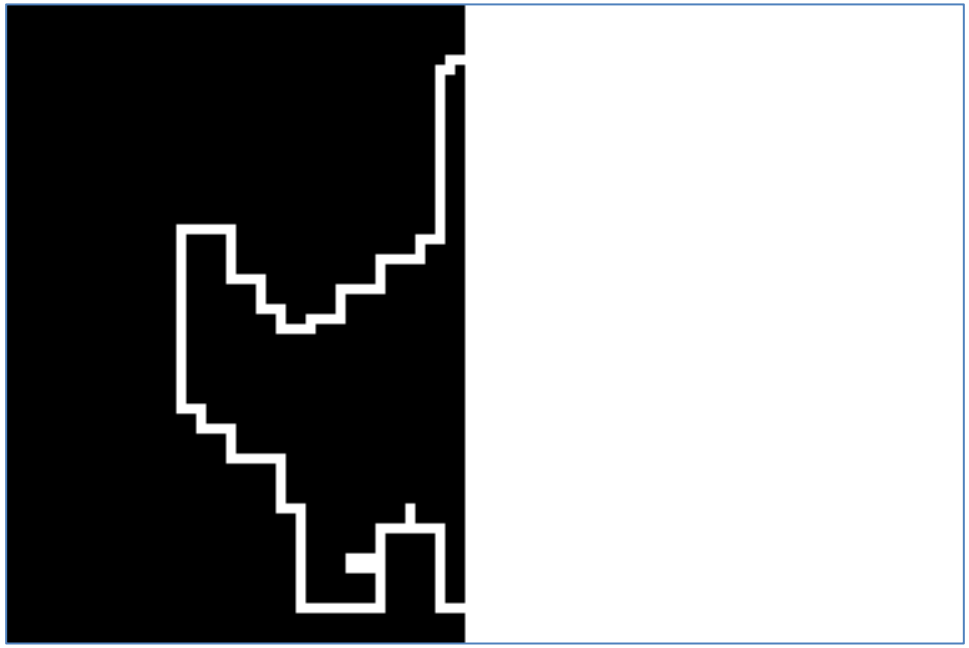
Опции

- **Без опций**.



Опции

- **DRAW_BLACK_NO** – отменить отрисовку черного цвета спрайта.

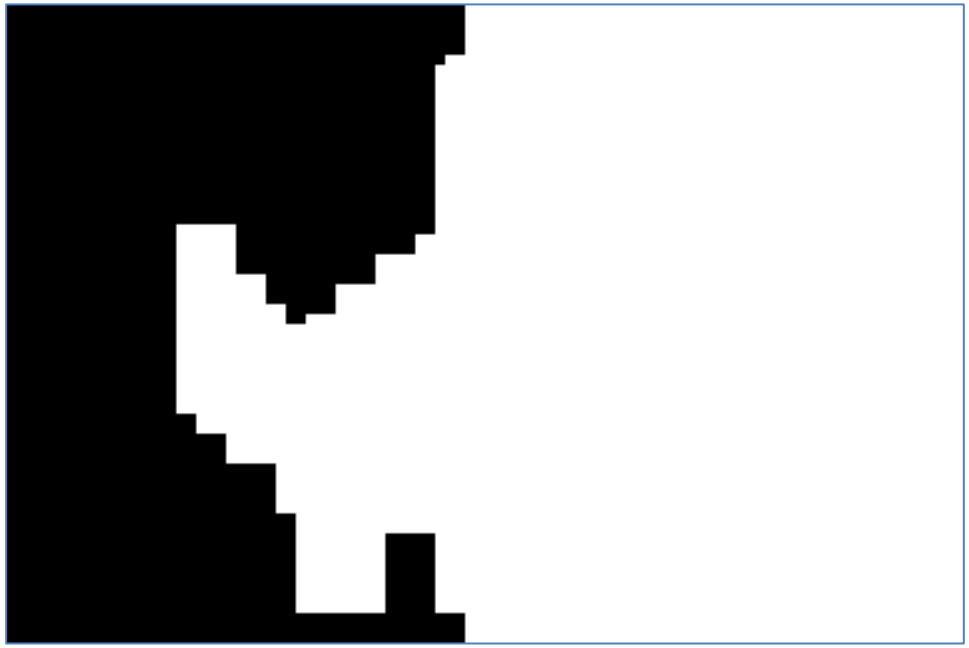


Опции

- **DRAW_WHITE_NO** – отменить отрисовку белого цвета спрайта.

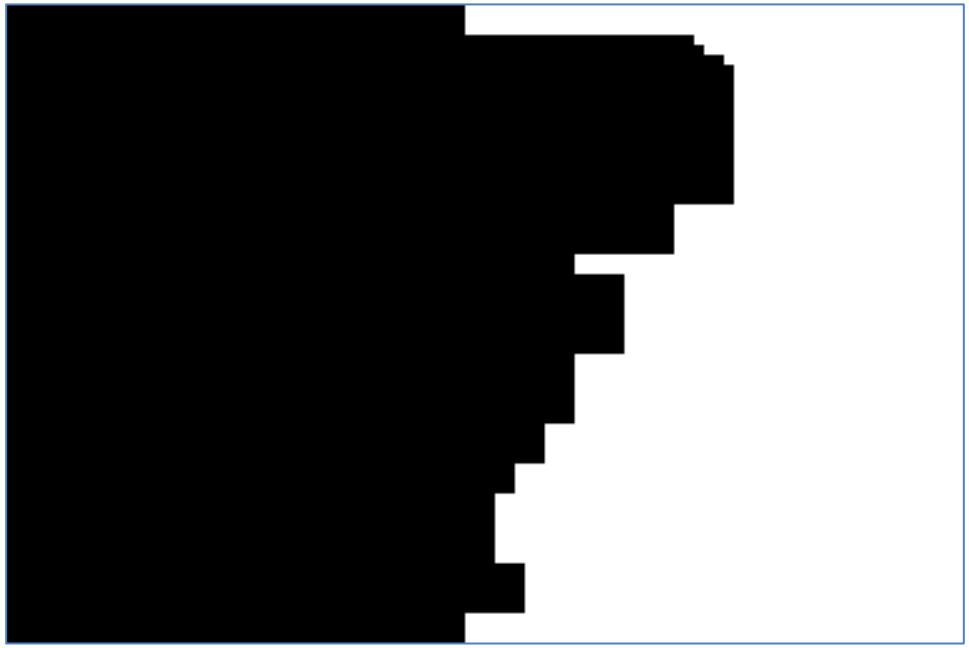
Опции

- **DRAW_BLACK_WHITE** – заменить черный цвет спрайта на белый.



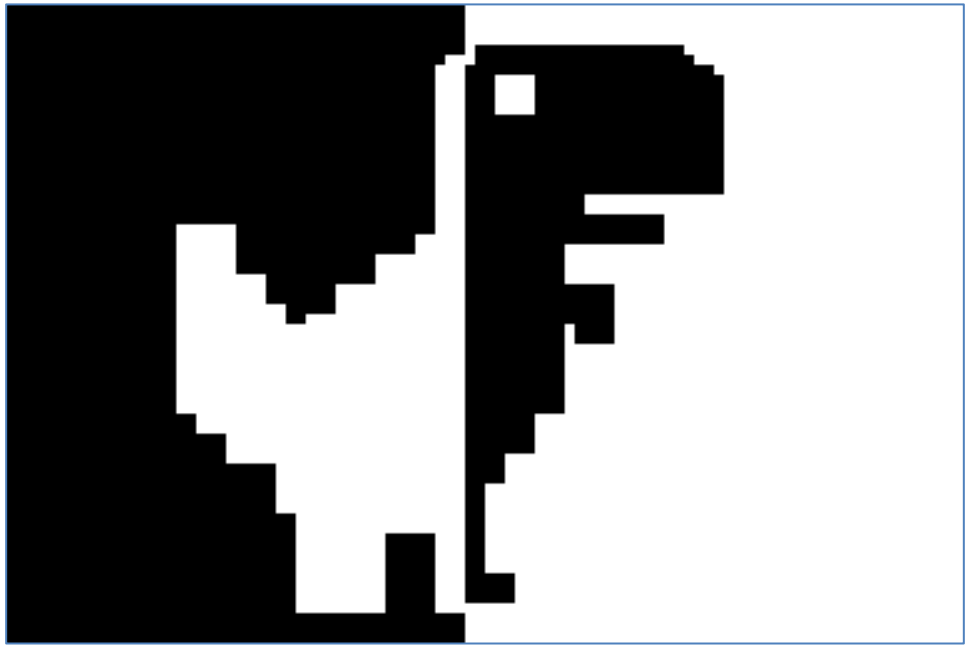
Опции

- **DRAW_WHITE_BLACK** – заменить белый цвет спрайта на черный.



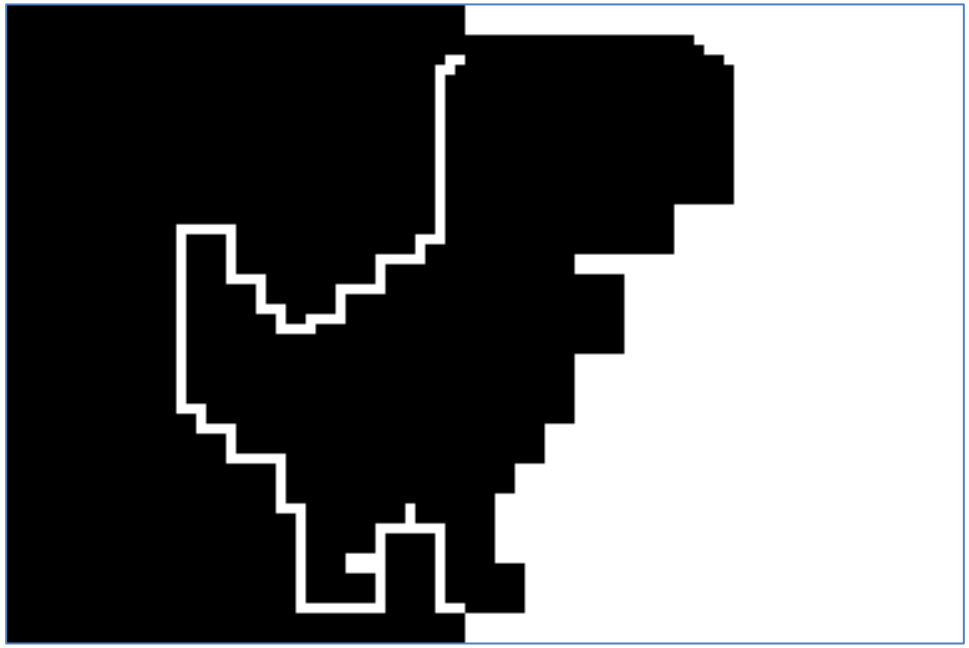
Опции

- **DRAW_BLACK_INVERT** – черный цвет инвертирует полотно.



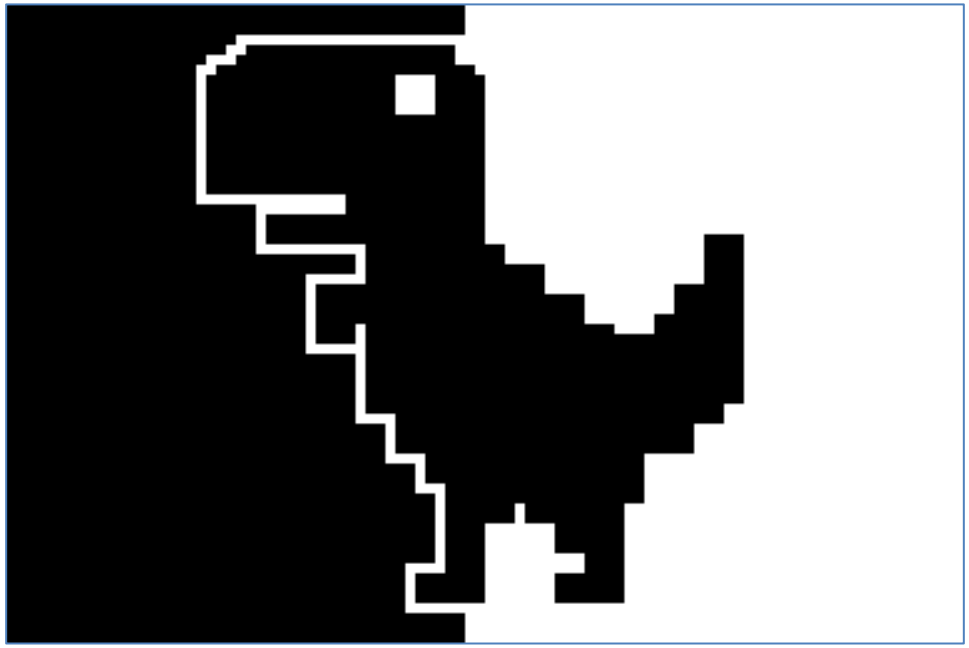
Опции

- **DRAW_WHITE_INVERT** – белый цвет инвертирует полотно.



Опции

- **DRAW_FLIP_SPRITE** – отразить весь спрайт вертикально.



Опции

- Опции можно использовать совместно, через битовый оператор ИЛИ.

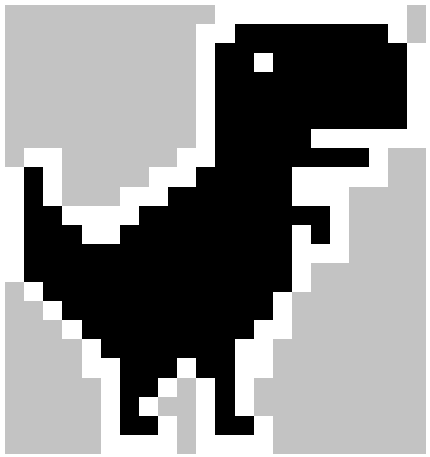
```
draw_sprite(dino, 0, x, y,
DRAW_FLIP_SPRITE|DRAW_WHITE_NO|DRAW_BLACK_INVERT);
```

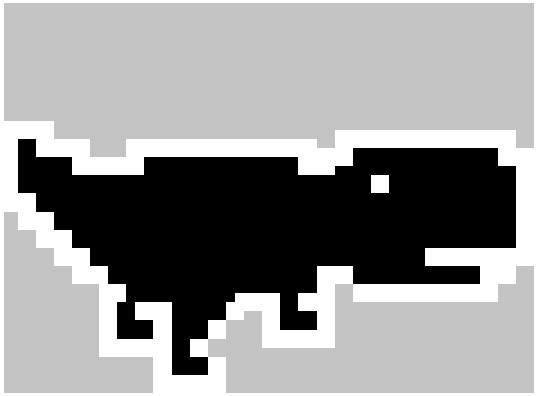
Пару слов о координатах

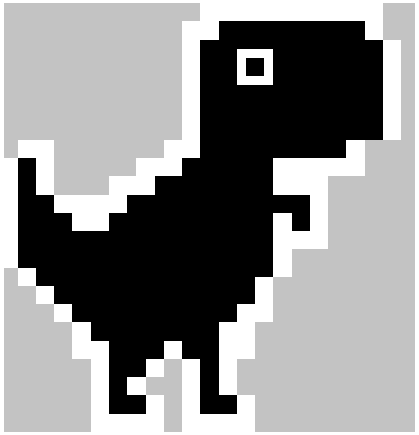
- Координаты дают вам возможность рисовать спрайты **на границе** экрана и даже **за границами** (правда тогда спрайт будет не видно)!

Костюмы

- Один спрайт – несколько костюмов!
  Выбирай любой при отрисовке.

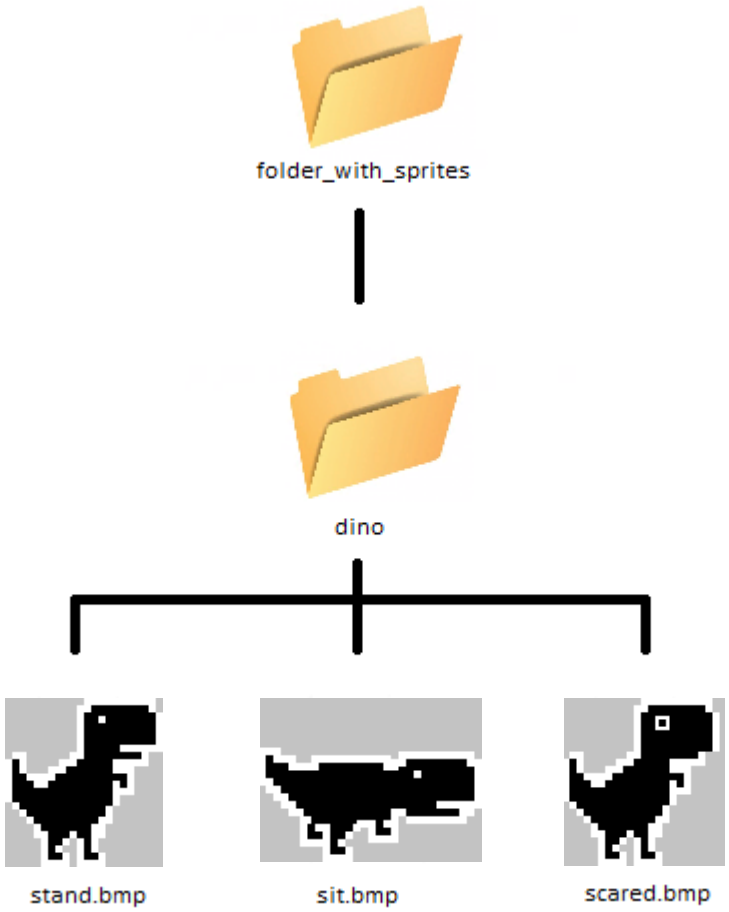






Костюмы

- Для этого в папке создаем еще одну вложенную папку, в которой несколько BMP картинок.
- Имя спрайта = имя папки.
- Имена костюмов = имена BMP файлов.



Костюмы

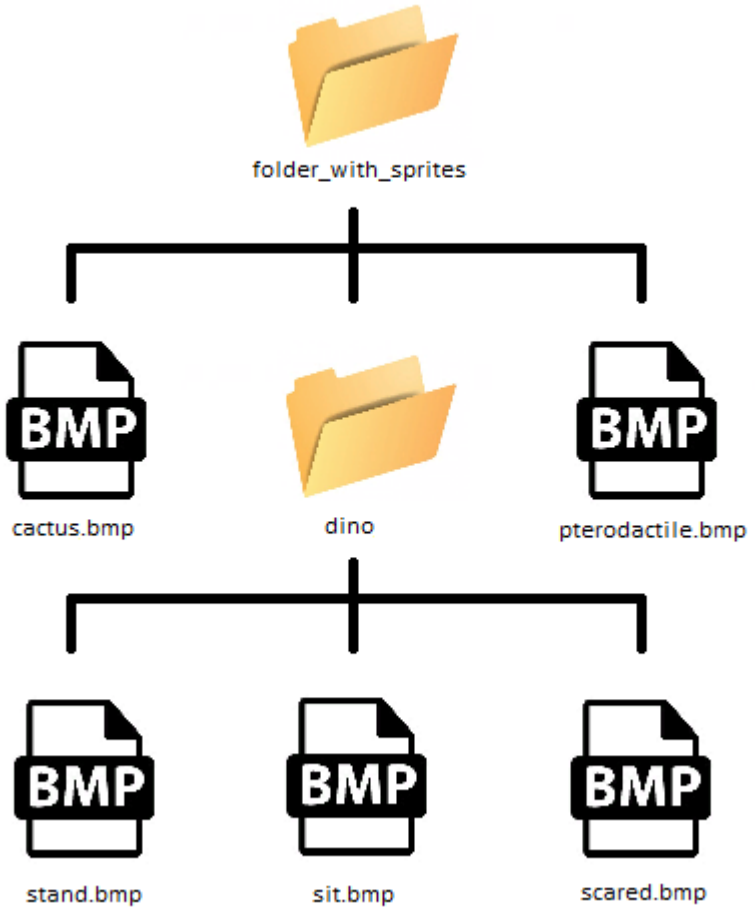
- Индексы костюмов лучше всего выбирать уже сгенерированные за Вас!
- Их осталось только импортировать!

```
extern uint32_t dino_stand, dino_sit, dino_scared;

draw_sprite(dino, dino_sit, x, y, 0);
```

Больше спрайтов!

- Создавайте много спрайтов в корневой папке, с костюмами или без.
- Скрипт превратит **все** в один C файл.



Проверка коллизии

Проверка коллизии со спрайтом

```
check_sprite_collision(dino, 0, x, y, 0);
```

- *Коллизия* = столкновение
- По умолчанию, коллизия = (*белый пиксель* спрайта пересекается с *белым пикселем* полотна ИЛИ **черный пиксель** спрайта пересекается с **черным пикселем**).
- Коллизия? – возващается `1`, иначе `0`.

Проверка коллизии

Опции:

- **CHECK_FLIP_SPRITE** – отразить спрайт при проверке
- **CHECK_BLACK_BLACK** – коллизия **черного** пикселя костюма с **черным** пикселем полотна. **CHECK_BLACK_WHITE** – коллизия **черного** пикселя костюма с **белым** пикселем полотна.
- **CHECK _WHITE_WHITE** – ... (по аналогии)
- **CHECK _WHITE_BLACK** – ...
- Опции точно так же можно комбинировать.

Детальная спецификация

bmp2arr

- Программа для преобразования массива картинок, сохранённого в *корневой директории*, в *C файл*, в котором записаны данные спрайтов.
- Должна работать на Linux
- `Usage: ./bmp2arr <dir_path> [output_file_path]`
- `dir_path` – путь к корневой директории.
- `output_file_path` – опциональное имя для C файла.
- C код программы `bmp2arr.c` не является частью кода для микроконтроллера.

Требования к корневой папке

- Корневая папка доступна для чтения
- Содержит только *объекты,* доступные для чтения
- *Объект* – Полные данные об одном спрайте.
- Объектом может быть
  - BMP файл (один костюм у спрайта),
  - Папка с BMP файлами (несколько костюмов).
- Прочих файлов в корневой папке нет.

При нарушении требований в консоли будет предупреждение/сообщ. об ошибке. Могут быть проигнорированы файлы или работа программы завершена без выхода С файла.

Требования к BMP файлу

- Доступен для чтения
- Имена BMP файлов (спрайтов и костюмов)
  - Длина имени – не более 50 символов
  - Допускается только латиница, цифры и «_».
  - **Имя спрайта** должно начинаться с буквы
- Сжатие – нет.
- Глубина цвета – 8 бит.
  - Возможно, изменится в будущем.

При нарушении требований в консоли будет соответствующее предупреждение, а сам файл будет проигнорирован.

Рисунок в BMP файле

- Черные пиксели (RGB = {0,0,0}) составляют *черный костюм спрайта.*
- Белые пиксели (RGB = {255,255,255}) составляют *белый костюм спрайта.*
- Остальные пиксели не влияют на костюмы спрайта. Они считаются «прозрачными».

Правила рисования костюма

- *Центром костюма* является верхний левый пиксель картинки.
- Если у спрайта несколько костюмов, то ширина и высота всех картинок не обязана совпадать.

Сгенерированный С файл

- Файл разделен на секции; в каждой описан один спрайт.
- В секции содержатся
  - данные черных и белых костюмов спрайта,
  - массив указателей на черные костюмы» и массив указателей на белые костюмы,
  - функция инициализации спрайта,
  - индексы костюмов (если их больше одного).

Сгенерированный С файл

- Никто не запрещает менять содержимое файла
- Изменения вносятся на свой страх и риск.

Структура Sprite

```
typedef struct {
      uint32_t width;
      uint32_t height;
      uint8_t **black_costumes_bank;
      uint8_t **white_costumes_bank;
      uint32_t n_costumes;
} Sprite;
```

Функция инициализации спрайта

```
void init_name_sprite(
      Sprite *sprite)
```

- Генерируется автоматически в С файле.
- ***name*** – заменяется на имя спрайта.
- Необходима перед использованием спрайта
- При использовании нужно экспортировать ключевым словом `extern`.

Нумерация спрайтов

- Генерируется автоматически в C файле при наличии более чем одного костюма у спрайта.
- Используется для обращения к необходимому костюму спрайта.
- При использовании нужно экспортировать ключевым словом `extern`.

Функция рисования спрайта

```
void draw_sprite(
      Sprite sprite,
      uint32_t costume,
      int16_t x,
      int16_t y,
      uint32_t options_mask)
```

Проинициализированный спрайт

Номер костюма спрайта

Координаты

Опции

- Координаты спрайта x и y считаются от верхнего левого пикселя экрана до *центра спрайта*.
- Координаты могут выводить спрайт внутри границ экрана, на границы и за них.

Функция рисования спрайта

- Не тратит ресурсы на отрисовку невидимых за пределами экрана пикселей.
- Сначала рисуется байт черного костюма, затем белого.

Функция рисования спрайта

Option	Description	Byte operation
DRAW_BLACK_NO	Cancels black costume rendering	~~`gmem_byte &= ~black_byte`~~
DRAW_WHITE_NO	Cancels white costume rendering	~~`gmem_byte |= white_byte`~~
DRAW_BLACK_WHITE	Draw `black_byte` in white color	`gmem_byte |= black_byte`
DRAW_WHITE_BLACK	Draw `white_byte` in black color	`gmem_byte &= ~white_byte`
DRAW_BLACK_INVERT	Invert color of `gmem_byte`	`gmem_byte ^= black_byte`
DRAW_WHITE_INVERT	Invert color of `gmem_byte`	`gmem_byte ^= white_byte`
DRAW_FLIP_SPRITE	Reverse order of black and white bytes in every row of the costume	-

- Опция **_NO** приоритетнее **_INVERT**
- Опция **_INVERT** приоритетнее замены цвета
- Опция **DRAW_FLIP_SPRITE** независима от остальных

Функция проверки спрайта на коллизию

```
uint32_t check_sprite_collision(
      Sprite sprite,
      uint32_t costume,
      int16_t x,
      int16_t y,
      uint32_t options_mask)
```

Проинициализированный спрайт

Номер костюма спрайта

Координаты

Опции

Функция проверки спрайта на коллизию

- Не проверяет коллизию невидимых за пределами экрана пикселей.
- Функция без аргументов проверки на коллизию всегда возвращает 0.
- Функция с аргументами взаимодополняющих проверок всегда вернет 1, если не весь спрайт за пределами экрана.

Функция проверки спрайта на коллизию

- Функция не отрисовывает спрайт, а лишь проверяет, произошла бы коллизия, если спрайт нарисовать с указанными параметрами.
- Проверка по аргументам идет в порядке:
  - **CHECK_BLACK_BLACK**
  - **CHECK_BLACK_WHITE**
  - **CHECK_WHITE_WHITE**
  - **CHECK_WHITE_BLACK**

Функция проверки спрайта на коллизию

Option	Description	Byte operation
CHECK_BLACK_BLACK	Check intersection of black pixels of sprite costume with black pixels in gmem	`if (~gmem_byte & black_byte) return 1`
CHECK_BLACK_WHITE	Check intersection of black pixels of sprite costume with white pixels in gmem	`if (gmem_byte & black_byte) return 1`
CHECK_WHITE_WHITE	Check intersection of white pixels of sprite costume with white pixels in gmem	`if (gmem_byte & white_byte) return 1`
CHECK_WHITE_BLACK	Check intersection of white pixels of sprite costume with black pixels in gmem	`if (~gmem_byte & white_byte) return 1`
CHECK_FLIP_SPRITE	Reverse order of black and white bytes in every row of the costume	-

- Опция **_NO** приоритетнее прочих
- Опция **CHECK_ FLIP_SPRITE** независима от остальных

Feedback

- Илья Левицкий: https://vk.com/id267671262
- Репозиторий: https://github.com/Levitsky-Ilya/stm32f0_ARM