ECS251 Project

The OS design for ECS 251

Team Members

Tzu-Hsin Yang (920227329), Yu-Cheng Hwang (918954206), Yan-Da Chen (920237445)

API List

▼ **All new added APIs are highlighted with red background**

Random Number Generator

- **genRandom() - generate a random number**

```
// Signature
#include "api.h"
int genRandom(int high);

// Parameters
int high - define the range from 0 to high (not inclusive)

// Return Value
return random_value
```

I/O Controller API

- **getStatus(): Listen to key inputs from keyboard**

```
// Signature
#include "api.h"
```

```
uint32_t getStatus(void);

// Parameters
void


// Return Value
return status
```

Mode Switch API

- **setTextMode() - change to text mode**

```
// Signature
#include "api.h"
void setTextMode(void);

// Parameters
void


// Return Value
no return value
```

- **setGraphicsMode() - change to graphics mode**

```
// Signature
#include "api.h"
void setGraphicsMode(void);

// Parameters
void


// Return Value
no return value
```

Text API

- **printLine() - Print a line of texts on screen in text mode**

```
// Signature
#include "api.h"
```

```
void printLine(char* string);

// Parameters
char* string - defines a string

// Return Value
no return value
```

Graphics API

- **setColor() - Set a specific color by assigning ARGB values to sprite palette**

```
// Signature
#include "api.h"
void setColor(int palette_id, int entry_id, uint32_t rgba);

// Parameters
int palette_id - defines which palette to assign color
int entry_id - defines which entry to assign color.
uint32_t rgba - defines the rgba value of color

// Return Value
no return value
```

- **setBackgroundColor() - Set a specific color by assigning ARGB values to background palette**

```
// Signature
#include "api.h"
void setBackgroundColor(int palette_id, int entry_id, uint32_t rgba);

// Parameters
int palette_id - defines which palette to assign color
int entry_id - defines which entry to assign color.
uint32_t rgba - defines the rgba value of color

// Return Value
no return value
```

- **calcSmallSpriteControl() - Calculate the 32 bit value of the small sprite controller**

```
// Signature
#include "api.h"
uint32_t calcSmallSpriteControl(uint32_t x, uint32_t y, uint32_t w, uint32_t h,
uint32_t p);

// Parameters
uint32_t x - defines the x coordinate of the left_upper corner of the rectangle.
uint32_t y - defines the y coordinate of the left_upper corner of the rectangle.
uint32_t w - defines the width of the rectangle.
uint32_t h - defines the height of the rectangle.
uint32_t p - defines the palette id that stores the color you want

// Return Value
return controller_value
```

- **calcLargeSpriteControl() - Calculate the 32 bit value of the large sprite controller**

```
// Signature
#include "api.h"
uint32_t calcLargeSpriteControl(uint32_t x, uint32_t y, uint32_t w, uint32_t h,
uint32_t p);

// Parameters
uint32_t x - defines the x coordinate of the left_upper corner of the rectangle.
uint32_t y - defines the y coordinate of the left_upper corner of the rectangle.
uint32_t w - defines the width of the rectangle.
uint32_t h - defines the height of the rectangle.
uint32_t p - defines the palette id that stores the color you want

// Return Value
return controller_value
```

- **calcBackgroundControl() - Calculate the 32 bit value of the background controller**

```
// Signature
#include "api.h"
uint32_t calcLargeSpriteControl(uint32_t x, uint32_t y, uint32_t z, uint32_t p);

// Parameters
uint32_t x - defines x position of the image
uint32_t y - defines y position of the image
uint32_t z - defines z position of the image
uint32_t p - defines the palette id that stores the color you want
```

```
// Return Value
return controller_value
```

- **setSmallSpriteControl() - Draw a rectangle with the small sprite controller**

```
// Signature
#include "api.h"
void setSmallSpriteControl(int sprite_id, uint32_t addr);

// Parameters
int sprite_id - defines id of sprite
uint32_t addr - defines the 32 bits to set the sprite

// Return Value
no return value
```

- **setLargeSpriteControl() - Draw a rectangle with the large sprite controller**

```
// Signature
#include "api.h"
void setLargeSpriteControl(int sprite_id, uint32_t addr);

// Parameters
int sprite_id - defines id of sprite
uint32_t addr - defines the 32 bits to set the sprite

// Return Value
no return value
```

- **setBackgroundSpriteControl() - Draw a rectangle with the background sprite controller**

```
// Signature
#include "api.h"
void setBackgroundSpriteControl(int sprite_id, uint32_t addr);

// Parameters
int sprite_id - defines id of sprite
uint32_t addr - defines the 32 bits to set the sprite

// Return Value
no return value
```

- **shiftSmallSpriteControl() - Shift controller to new coordinates**

```
// Signature
#include "api.h"
void shiftSmallSpriteControl(int sprite_id, uint32_t x, uint32_t y);

// Parameters
int sprite_id - defines id of sprite
uint32_t x - defines the x coordinate expected to shift to
uint32_t y - defines the y coordinate expected to shift to

// Return Value
no return value
```

- **shiftLargeSpriteControl() - Shift controller to new coordinates**

```
// Signature
#include "api.h"
void shiftLargeSpriteControl(int sprite_id, uint32_t x, uint32_t y);

// Parameters
int sprite_id - defines id of sprite
uint32_t x - defines the x coordinate expected to shift to
uint32_t y - defines the y coordinate expected to shift to

// Return Value
no return value
```

- **getSmallSpriteControl() - Get the 32 bit value of the controller**

```
// Signature
#include "api.h"
uint32_t getSmallSpriteControl(int sprite_id);

// Parameters
int sprite_id - defines id of sprite

// Return Value
return the stored 32-bit value
```

- **getLargeSpriteControl() - Get the 32 bit value of the controller**

```
// Signature
#include "api.h"
```

```
uint32_t getLargeSpriteControl(int sprite_id);

// Parameters
int sprite_id - defines id of sprite

// Return Value
return the stored 32-bit value
```

- **getBackgroundSpriteControl() - Get the 32 bit value of the controller**

```
// Signature
#include "api.h"
uint32_t getBackgroundSpriteControl(int sprite_id);

// Parameters
int sprite_id - defines id of sprite

// Return Value
return the stored 32-bit value
```

Multithreading

- **initContext() - Create a new thread**

```
// Signature
#include "api.h"
TContext initContext(uint32_t *stacktop, TEntry entry, void *param);

// Parameters
uint32_t *stacktop
TEntry entry
void *param

// Return Value
return TContext
```

- **switchContext() - Switch threads**

```
// Signature
#include "api.h"
void SwitchContext(TContext *old, TContext new);

// Parameters
TContext *old - defines the thread that are original thread
```

```
TContext new - defines the thread that are next thread to be run on

// Return Value
no return value
```

Timer API

- **getTicks() - Get the current time**

```
// Signature
#include "api.h"
uint32_t getTicks(void);

// Parameters
void

// Return Value
return global
```

Video Interrupt

- **getVideoInterruptCount() - Get the number of occurrence of video interrupt**

```
// Signature
#include "api.h"
int getVideoInterruptCount(void);

// Parameters
void

// Return Value
return video_interrupt_count
```

CMD Interrupt

- **getCMDInterruptCount() - Get the number of occurrence of CMD interrupt**

```
// Signature
#include "api.h"
int getCMDInterruptCount(void);

// Parameters
```

```
void

// Return Value
return cmd_interrupt_count
```

Memory Management

- **malloc() - Allocates the requested memory and returns a pointer to it**

(only implemented on cartridge side, not as an API)

```
// Signature
#include "memory.c"
void* malloc(size_t size)

// Parameters
int size - the required memory size

// Return Value
A void pointer to the allocated space, or NULL if there's insufficient memory
available.
```