

Story

The game takes place a number of years after the events of EoS.D. Flandre's 500th birthday approaches, and the SDM residents prepare for a centennial celebration. However, Flandre is still accosted by night terrors that prevent her from feeling well enough for the occasion. The player must guide Flandre through the overworld, unlocking dream worlds, and clearing them, in order to defeat the terrors within.

Technical notes

Kouma uses cocos2d/GLFW, which does not support selecting a GPU device. If you are using a multi-GPU system (meaning a "GPU 0" and "GPU 1" in the Windows 10 task manager), you will need to use Nvidia/AMD control panel and select to use dedicated device for kouma.exe, although the game should be playable on most integrated graphics.

The game engine technically supports running at any resolution and framerate. Mostly, it has been tested at 1080p60, although it has also been tested at 720p, and at 30 FPS as well as 120 FPS.

Kouma supports gamepads. In the output log, you should see a message "gamepad connected" or "gamepad not connected". It supports custom control mappings for both the gamepad and keyboard.

Kouma uses OpenAL for the sound engine. It should support surround sound output automatically based on the Windows sound device configuration.

Kouma uses dynamic lighting. The lighting currently doesn't have any gameplay effect, other than affecting what the player can see. (The only exception to this are darkness area rooms, where the player takes damage if all torches out go out.) Generally, darker areas indicate optional areas of greater difficulty.

Application data is stored in "%USERPROFILE%\AppData\Local\Koumachika\". This directory should be automatically created when the application is launched. In turn, it contains "replays" and "profiles" directories, as well as log.txt. Here, you can create config.txt to configure application options.

Gameplay concepts

Equip slots - The UI shows four equip slots in a plus-shape. The idea is to have an equip slot for each kind of action (fire, spell, bomb, power attack), and use the corresponding dpad button to cycle that equip slot. The demo does not have additional bombs or power attacks. Instead, the extra two buttons are used to allow cycling through spells and fire patterns bi-directionally.

Profile/savegame – The game provides 9 save slots, corresponding to profile1.profile – profile9.profile. The corresponding profile name is selected when choosing a new game. Technically, the game engine supports loading and saving profiles with any file name; however, when you choose to save from within the game, it will save to the same file name that was loaded, or profile1-9 for the new game first save.

Note0: The game includes autosaving: *autosave.profile* will be created/overwritten after completing a level. It is not recommended to load *autosave.profile* directly (see below).

Note1: in *log.txt*, a confirmation will be printed when a profile is loaded, saved, or overwritten.

Note2: Since the game engine uses the same file that was loaded as the save location, you probably don't want to load *autosave.profile*. Instead, to recover progress from the autosave, simply copy or rename it to a different file name, including replacing an existing profile1-9. If you want, you should preview the *autosave.profile* in the Load Profile menu before copying it to a profile slot.

Replay - The game automatically captures replay data when a level is being played. The replay will automatically be saved if the level is completed. The player also has the option to save the replay in the game over menu. The replay does not process UI input, such as pausing the game, or advancing dialog. Instead, live controls are used. The game engine must be running at the same frame rate at which the replay was created. Currently, unlike save profiles, there isn't a preview menu to show replay details of the selected file before running it. So, if the framerate doesn't match, it will simply fail to load, and an error message will be printed to *log.txt*.

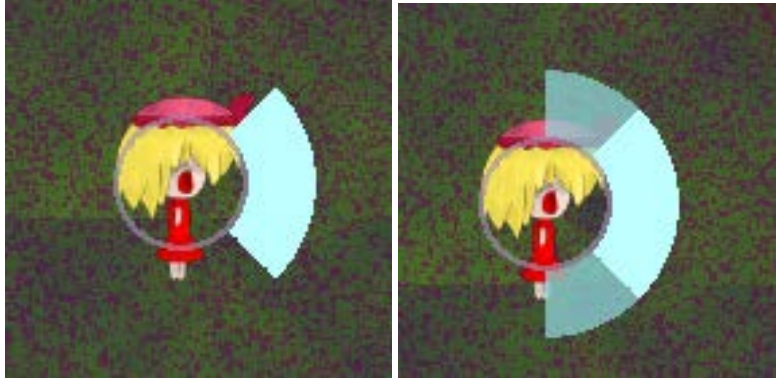
Dream World – As Flandre progresses through the world, she unlocks dream worlds, where she can violently confront her fears. Dream worlds can be activated from the pause menu in the overworld, or by interacting with the desk in Flandre's room.

In the dream world, Flandre must fight enemies and reach the goal in the current level. When Flandre interacts with the goal, the level is completed. If the goal is in a boss room, the enemy must be defeated before it can be used.

If Flandre reaches 0 HP in the dream world, it is game over. All progress in the current level is lost, except when a floor is successfully completed. The player can save in the overworld, or between floors.

A world consists of multiple floors, about 4-6. When a level is completed, it unlocks the next level in that world. New worlds are only unlocked via progress in the overworld. Clearing floors also has effects on the overworld, and not just the final levels of each world; look for new things appearing, and even obstacles disappearing in the overworld. Only the first world (Graveyard) is fully implemented.

Focus/shield – In focus mode, you will be able to see your own physics body (“hit circle”), as well as your graze radius and shield area. Your movement speed will be reduced, but the player's shield will be active. When in focus mode, bullets that hit the player front-on will not damage her, although blocking bullets will drain power; shield does not protect against touch damage, bombs, or area-of-effect spells. At a shield level of 1, the player will only be able to deflect bullets within 45 degrees of the angle they are facing. Each upgrade adds another 45 degree slice of protection, and reduces the stamina cost for angles that were previously covered. Focus mode can also be used to determine the exact angle the player is facing.



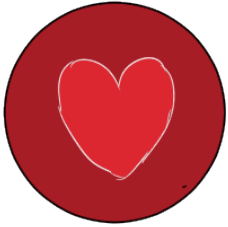
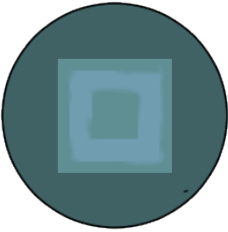
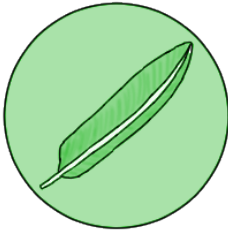
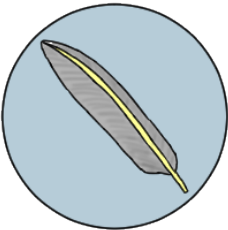
Shield level	0-45 deg	45-90 deg	90 – 135 deg	135-180 deg
1	10 stamina	n/a	n/a	n/a
2	5 stamina	10 stamina	n/a	n/a
3	2.5 stamina	7.5 stamina	12.5 stamina	n/a
4	0 stamina cost	5 stamina	10 stamina	15 stamina


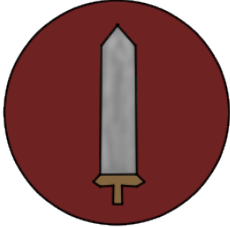
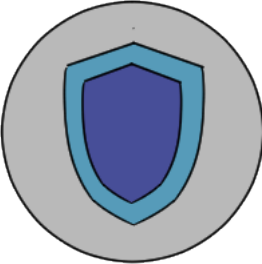
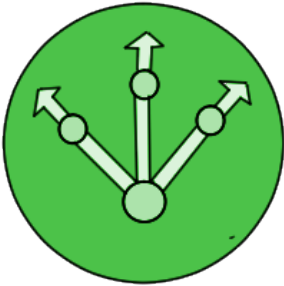
Combo – combo points are acquired every time you damage an enemy. They drain over time, in addition to being zeroed out if you take damage. If you get five hits a second, you will reach 100% combo in five seconds. Once combo mode is active, it will stay active until combo point reach zero, or until you take damage. When active, the player will have a gold flicker; combo gives the player double graze reward, and a 25% attack damage bonus.

Bomb – The player can summon bombs, which materialize in front of her. If you press bomb with focus/shield active, she will throw the bomb instead. Materializing a bomb uses 5 MP. Bombs will do full damage to enemies within half of their blast radius, and the damage will drop off to zero at the edge of the radius. Bombs also create a large knockback force. Bombs can break breakable walls. Bombs can be bumped into, by either the player or enemies, and move across the floor.

Upgrades

Upgrades change Flandre's attributes in the dream world, though they can also be found in the overworld. When an upgrade is found in a dream world level, it is applied immediately, but it will be lost if you quit the level or get a game over before clearing the floor.

	Max HP	Increase the player's max HP by 25 (as well as current HP).
	Max MP	Increase the player's max MP by 25 (as well as current MP).
	Max stamina	Increases max stamina by 25 (as well as current stamina).
	Agility	Increase agility by 1 point. Agility is applied exponentially, with each point increasing speed by 33%

 A green sword icon centered within a dark green circular background.	Attack speed	Increases the rate at which attacks are performed, including the fire rate from fire patterns and power attack (Lavaeteinn).
 A grey sword icon centered within a dark red circular background.	Attack damage	Increases attack (bullet damage) by 25%.
 A shield icon with a blue border and a purple center, centered within a light grey circular background.	Shield Level	Grants or upgrades the shield, which allows the player to block attacks in focus mode. Upgrading shield will increase the angle at which bullets can be blocked, and reduce cost of blocking bullets front-on.
 Three white arrows pointing upwards and outwards from a central point, centered within a green circular background.	Bullet speed	Adds speed multiplier to player bullets, 50% per upgrade.

Controls

Gamepad:

Left stick to move

Right stick to aim (and fire if auto-fire)

L3 – center look – the player will face the direction of their movement. Holding it for 0.5 seconds toggles auto-look mode, where the player will continue to face the direct of movement input. If right stick is pressed, it will still override auto-look

R3 – toggle auto-fire. (i.e. the player will automatically fire when right stick is pressed, if applicable)

A – interact, menu select

X – Spell

B – bomb, menu back

Y – power attack (melee)

Start – Pause

Back – map (also pauses the game)

D-Pad – d-pad directions correspond to the equip slot of the corresponding right thumb button and action. Ideally dpad-up cycles through power attacks, right changes bomb equip, left spells, and down fire patterns. However, since the game currently does not have any alternate power attacks or bombs, left/right is used to cycle spells, and up/down to cycle through fire patterns in either direction.

L1/R1 –dodge

L2 – focus/shield mode

R2 – fire

Keyboard:

WASD: move

arrows : aim

escape - pause, menu back

enter – interact, menu select

E: interact, menu select

X: dialog skip, center look

M: map menu

Z: toggle auto fire

1/2 – spell previous/next

3/4 – fire pattern previous/next

Q: spell
C: bomb
space: power attack

ALT (left or right) - dodge
SHIFT (left or right) – shield/focus
CTRL (left or right) – fire

config.txt & custom controls

App configuration can be performed in "%HOMEPATH%\AppData\Local\Koumachika\config.txt". The app data directory should be created when the application is first run, though you will need to create the config file.

The config file should contain one directive per line, and tokens should be space-separated. Directives are processed by their order in the file. Empty lines are ignored, as are lines that start with a hash symbol or two slashes (“#” or “//”). If a setting doesn’t seem to work, check log.txt for error messages. Note that directives are case-sensitive, but boolean words (“true”, “false”), control actions, button names & key names are case-insensitive.

The following directives are recognized in the config file:

`framerate [X]` – must be a positive integer value. Default 60
`fullscreen [bool]` – must be a Boolean value, such as “true” or “false” (case-insensitive) or “0” or “1”, default false
`multithread [bool]` – must be a Boolean value, default true. This option causes the game logic (physics engine & object update) to run in a separate thread. Recommended if you are running the game at a high rate (120+FPS).
`resolution [width] [height]` – must be positive integer values, default 1600x1000.
`vsync [bool]` – default true. Recommended if you are running the game fullscreen at the same frame rate as the monitor; it is recommended to leave multithread turned on for this.
`difficulty [scale]` – Must be a floating point (or integer) value 0.25 – 4.0, default 1.0. This scales the damage the player receives, and the inverse for enemies, i.e. lower value is lower difficulty.

config.txt can also contain control mapping directives:

`key [*] [...] / button [*] [...]` – Assigns a key/button one or more actions, i.e. the key or button name, and then a space separated list of actions. This will overwrite the default control mapping assigned to this key/button, as well as a previous assignment to this key/button. Generally, a

key/button should only be assigned one gameplay action; the intended use for mapping a key/button to multiple actions is to allow independently setting gameplay and menu/UI control bindings.

```
#example: keyboard southpaw, swapping WASD and arrow keys
key W aim_up
key S aim_down
key A aim_left
key D aim_right
```

```
key RIGHT_ARROW move_right
key LEFT_ARROW move_left
key DOWN_ARROW move_down
key UP_ARROW move_up
```

```
#example: swap X and B (spell/bomb), but keep B as menu_back:
button X bomb
button B spell menu_back
```

`clear_key [*] / clear_button [*]` – Removes the assignment to a given keyboard key or gamepad button. It isn't necessary to use this before assigning, only if you want to un-map unused buttons (most likely on the keyboard) against accidental presses.

```
clear_key A
clear_button L1
```

`southpaw [bool]` - If set, the gamepad left stick will aim and the right stick will move the player. note that this has no effect on the `move_*` or `aim_*` bindings, only gamepad control sticks.

`clear_all_keys` – clear all keyboard key bindings

`clear_all_buttons` – clear all gamepad button bindings

The following key values are recognized:

ESCAPE	LEFT_ARROW	APOSTROPHE	LEFT_BRACKET	N
BACKSPACE	RIGHT_ARROW	COMMA	BACK_SLASH	O
TAB	UP_ARROW	MINUS	RIGHT_BRACKET	P
RETURN	DOWN_ARROW	PERIOD	BACKTICK	Q
SHIFT	F1	SLASH	A	R
LEFT_SHIFT	F2	0	B	S
RIGHT_SHIFT	F3	1	C	T
CTRL	F4	2	D	U
LEFT_CTRL	F5	3	E	V
RIGHT_CTRL	F6	4	F	W
ALT	F7	5	G	X
LEFT_ALT	F8	6	H	Y
RIGHT_ALT	F9	7	I	Z
HOME	F10	8	J	ENTER
PG_UP	F11	9	K	
END	F12	SEMICOLON	L	
PG_DOWN	SPACE	EQUAL	M	





The button names match the convention of the libgainput naming, which uses a mixed convention for naming controller buttons: right-thumb buttons are described using Xbox convention (A/B/X/Y), whereas the other buttons L1/R1, L3/R3, etc are described using PlayStation control convention. The control sticks are handled as a special case, and currently cannot be re-mapped. move_* and aim_* actions are intended for keyboard use, although they can technically be used to map d-pad or other buttons to movement or aiming.


Start	Up	X	L2
Select	Down	Y	R2
Left	A	L1	L3
Right	B	R1	R3

Control actions:

pause		
map_menu	fire_pattern_previous	move_up
menu_select	fire_pattern_next	move_right
menu_back	spell_previous	move_down
dialog_skip	spell_next	move_left
interact	sprint	aim_up
fire	focus	aim_right
spell	center_look	aim_down
bomb	fire_mode	aim_left
power_attack		

Enemies

 A chibi-style character with short, wavy yellow hair, wearing a red dress with light blue wings on the shoulders.	Red Fairy	Red fairy – Slower and with more HP. They can throw bombs at the player, and also have bomb resistance.
 A chibi-style character with short, wavy yellow hair, wearing a green dress with light blue wings on the shoulders.	Green Fairy	High agility enemy, with ability to evade projectiles. They stress from being shot at and will return fire.
 A chibi-style character with short, wavy yellow hair, wearing a blue dress with light blue wings on the shoulders.	Blue Fairy	They will usually follow a set path in the room they are in. They have a shield similar to the player, making it difficult to defeat them with frontal hits. They have a radius power attack that it is activated if the player gets in front of them.
 A chibi-style character with short, wavy teal hair, wearing a purple dress with light blue wings on the shoulders.	Zombie Fairy	Slow, with heavy darkness touch damage. They also have a magic effect that will also put out nearby torches.

	Bat	High agility, but only threat is touch damage.
---	-----	--

License / Copyrights

This project includes sprites from Touhou 6: EoSD and PatchCon: DtL (https://www.spriter-resource.com/pc_computer/touhoukoumakyouthembodimentofscarletdevil/sheet/33553/) (https://www.spriter-resource.com/pc_computer/patchcon/sheet/23185/)

This project includes sound effects from Touhou and <https://freesfx.co.uk/>.