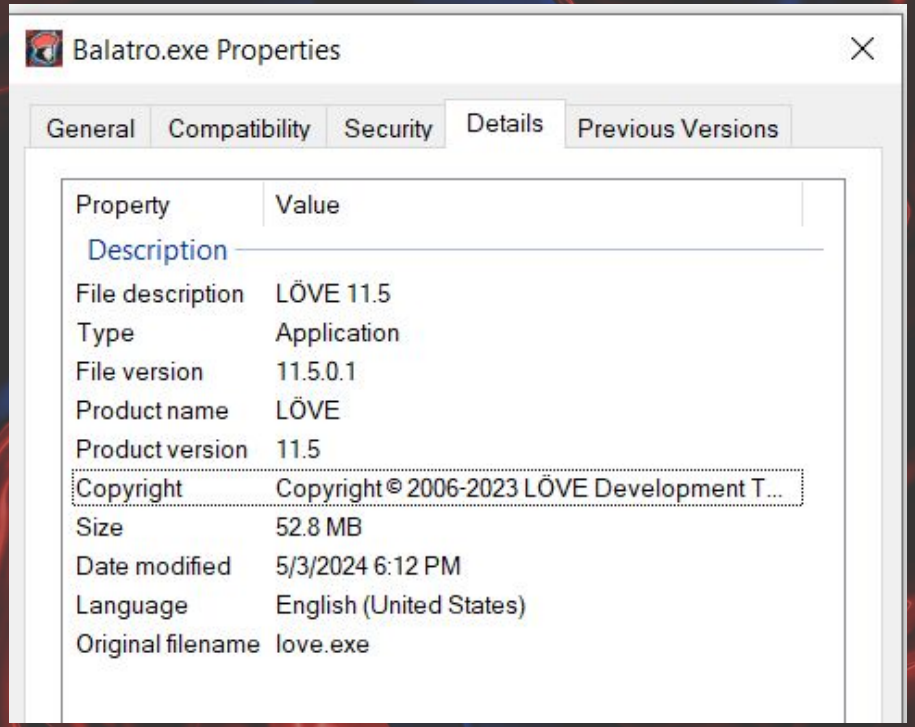
BALATRO

Luke Severs, Jacob Sharp, Camp Steiner

Planning the Heist

- Reverse Engineer Balatro
- Find out how it stores jokers, gameplay, and cards
- Make our own mods that tamper with these

Balatro.exe


Balatro.exe Properties
General
Compatibility
Security
Details
Previous Versions
Property
Value
Description
File description
LÖVE 11.5
Type
Application
File version
11.5.0.1
Product name
LÖVE
Product version
11.5
Copyright
Copyright © 2006-2023 LÖVE Development T...
Size
52.8 MB
Date modified
5/3/2024 6:12 PM
Language
English (United States)
Original filename
love.exe

Balatro.exe

```
// attributes: thunk
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  return WinMain_0(hInstance, hPrevInstance, lpCmdLine, nShowCmd);
}
```

Balatro.exe

```
int __stdcall WinMain_0(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  const WCHAR *CommandLineW; // rax
  LPWSTR *v5; // r12
  SIZE_T v6; // rbx
```

...

```
  }
  v9[v11] = 0i64;
  LocalFree(v5);
  SDL_SetMainReady();
  v19 = setup_lua_runtime(pNumArgs, v9);
  if ( pNumArgs > 0 )
  {
    v20 = v9;
    do
    {
      v21 = GetProcessHeap();
      HeapFree(v21, 0, *v20);
      ++v10;
      ++v20;
    }
    while ( v10 < pNumArgs );
  }
  v22 = GetProcessHeap();
  HeapFree(v22, 0, v9);
  return v19;
}
```

Balatro.exe

```
37  while ( v5 != 5 );
38  while ( argc <= 1 || strcmp(argv[1], "--version") )
39  {
40    lua_State = (love *)luaL_newstate();
41    luaL_openlibs((__int64)lua_State);
42    lua_getfield(lua_State, 0xFFFFD8EEi64, "package");
43    lua_getfield(lua_State, 0xFFFFFFFFi64, "preload");
44    lua_pushcclosure(lua_State, &luaopen_love_jitsetup, 0i64);
45    lua_setfield(lua_State, 0xFFFFFFFEi64, "love.jitsetup");
46    lua_settop(lua_State, 0xFFFFFFFDi64);
47    lua_getfield(lua_State, 0xFFFFD8EEi64, "require");
48    lua_pushstring(lua_State, "love.jitsetup");
49    lua_call(lua_State, 1i64, 0i64);
50    lua_getfield(lua_State, 0xFFFFD8EEi64, "package");
51    lua_getfield(lua_State, 0xFFFFFFFFi64, "preload");
52    lua_pushcclosure(lua_State, &luaopen_love, 0i64);
53    lua_setfield(lua_State, 0xFFFFFFFEi64, "love");
54    lua_settop(lua_State, 0xFFFFFFFDi64);
55    lua_createtable(lua_State, 0i64, 0i64);
56    if ( argc > 0 )
57    {
58      lua_pushstring(lua_State, *argv);
59      lua_rawseti(lua_State, 0xFFFFFFFEi64, 0xFFFFFFFEi64);
60    }
61    lua_pushstring(lua_State, "embedded boot.lua");
62    lua_rawseti(lua_State, 0xFFFFFFFEi64, 0xFFFFFFFFi64);
63    v8 = 1;
64    if ( argc > 1 )
65    {
66      v9 = argv + 1;
67      do
68      {
69        lua_pushstring(lua_State, *v9);
70        lua_rawseti(lua_State, 0xFFFFFFFEi64, (unsigned int)v8++);
71        ++v9;
72      }
73      while ( v8 < argc );
74    }
75    lua_setfield(lua_State, 0xFFFFD8EEi64, "arg");
76    lua_getfield(lua_State, 0xFFFFD8EEi64, "require");
77    lua_pushstring(lua_State, "love");
78    lua_call(lua_State, 1i64, 1i64);
79    lua_pushboolean(lua_State, 1i64);
80    lua_setfield(lua_State, 0xFFFFFFFEi64, "_exe");
81    lua_settop(lua_State, 0xFFFFFFFEi64);
82    lua_getfield(lua_State, 0xFFFFD8EEi64, "require");
      lua_pushstring(lua_State, "love.boot");
```

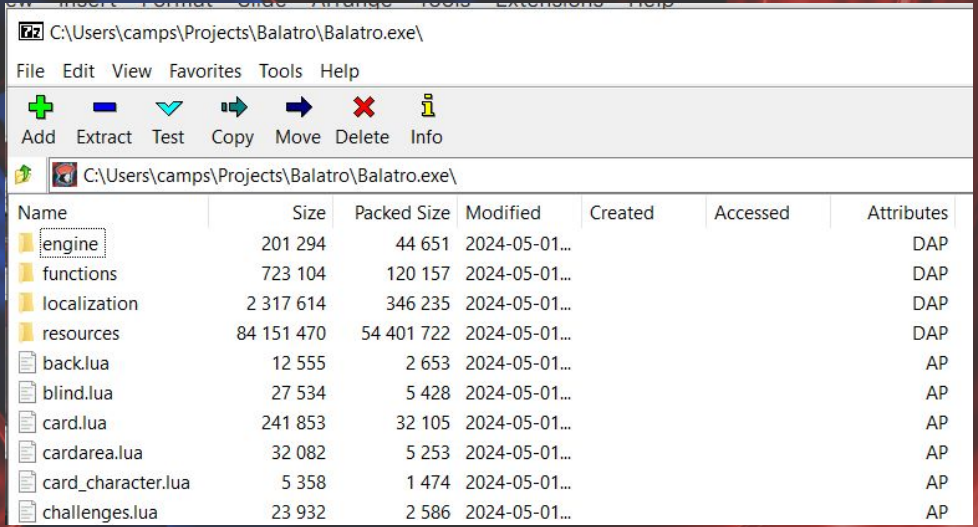
Lua and LÖVE

- Written in Lua
- JIT-interpreted scripting language
- Executable does little other than initialize Lua runtime and call necessary files and functions
- LÖVE engine handles actual gameplay

Lua and LÖVE

- Pros
  - Can easily extract source files
  - Lua is a very simple language to work with

- Cons
  - Less is handled by the executable
  - More abstraction between us and the code we need to modify

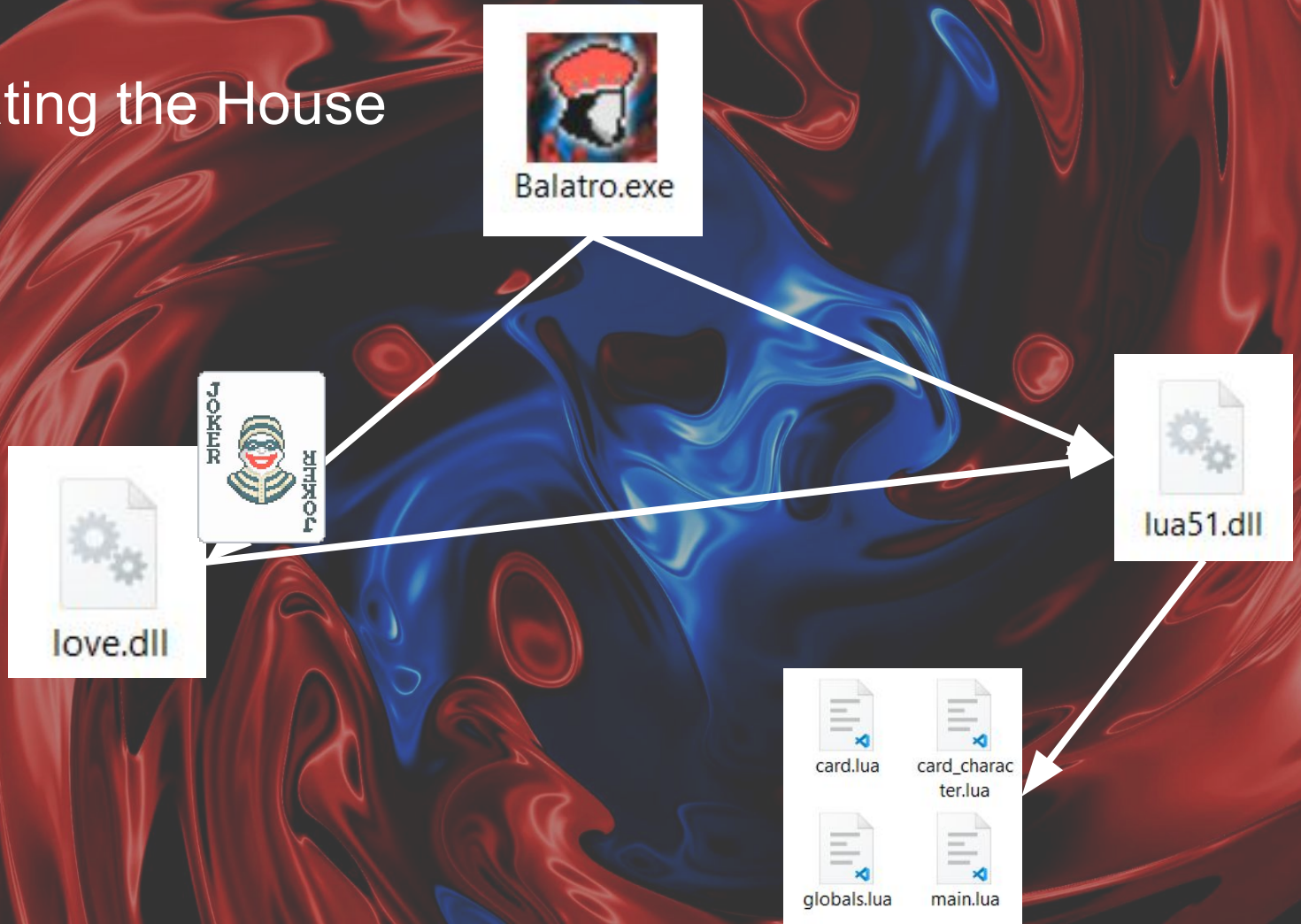


Name	Size	Packed Size	Modified	Created	Accessed	Attributes
engine	201 294	44 651	2024-05-01...			DAP
functions	723 104	120 157	2024-05-01...			DAP
localization	2 317 614	346 235	2024-05-01...			DAP
resources	84 151 470	54 401 722	2024-05-01...			DAP
back.lua	12 555	2 653	2024-05-01...			AP
blind.lua	27 534	5 428	2024-05-01...			AP
card.lua	241 853	32 105	2024-05-01...			AP
cardarea.lua	32 082	5 253	2024-05-01...			AP
card_character.lua	5 358	1 474	2024-05-01...			AP
challenges.lua	23 932	2 586	2024-05-01...			AP

Lua and LÖVE


Balatro.exe
Setup LÖVE Runtime and JIT compiler
Setup Lua Environment
Run Lua
love.dll
lua51.dll
Load Lua files
card.lua
card_charac ter.lua
globals.lua
main.lua

Beating the House



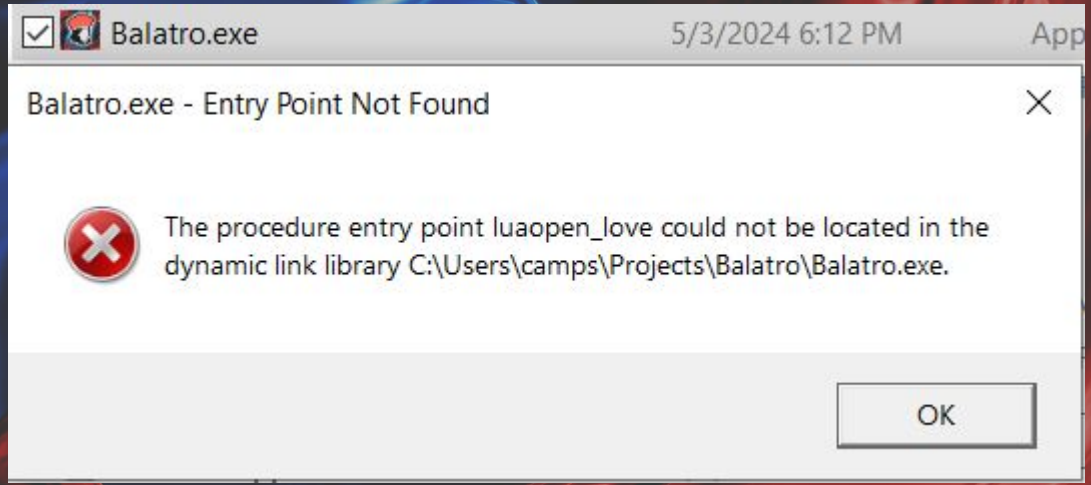
Beating the House

- Windows scans for DLLs by name, starting in the directory the program is located in
- Write our own love.dll that does all of the functionality of the real love.dll plus a little extra
- Could also do the same for lua51.dll and replace the load function

```
// imports in Balatro.exe from love.dll
__declspec(dllexport) __int64 luaopen_love();
__declspec(dllexport) char* love_codename(void);
__declspec(dllexport) const char* love_version(void);
__declspec(dllexport) __int64 luaopen_love_jitsetup(__int64);
__declspec(dllexport) char love_openConsole(const char **);

__declspec(dllexport) BOOL DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved) {
    OutputDebugStringW(L"Hello, World from DllMain!");
    if ( fdwReason == 1 )
        DisableThreadLibraryCalls(hinstDLL);
    return 1;
}

__declspec(dllexport) int love::luax_resume(void*, void*, int, int*) {
    return 0;
```

House always wins

- Skill Issue



Beating the House

Beating the House

- What if we make our own “Balatro.exe” that calls LÖVE setup and Lua setup and also calls some extra stuff

```cpp
typedef __int64 (*LuaOpenLoveFunction)(__int64);
typedef char* (*LoveCodenameFunction)();
typedef int (*LoveLuaxResumeFunction)(void*, void*, int, int*);
typedef const char* (*LoveVersionFunction)();
typedef __int64 (*LuaOpenLoveJitSetupFunction)(__int64);
typedef char (*LoveOpenConsoleFunction)(const char**);

int main(int argc, char** argv) {

    #pragma region Load DLL
    HMODULE lovedll = LoadLibraryA("C:/Users/camps/Projects/Balatro/love.dll");

    if (lovedll == NULL)
    {
        std::cerr << "Failed to load DLL" << std::endl;
        return 1;
    }

    LuaOpenLoveFunction luaopen_love = (LuaOpenLoveFunction)GetProcAddress(lovedll, "luaopen_love");
    if(luaopen_love == NULL)
    {
        std::cerr << "Failed to locate luaopen_love function" << std::endl;
        FreeLibrary(lovedll);
        return 1;
    }
```

```cpp
  const char* v4; // rax
  __int64 v5; // r9
  char v6; // r8
  struct lua_State *lua_State; // rbx
  int v8; // edi
  _QWORD *v9; // rsi
  int v10; // edi
  int v11; // eax
  unsigned int v12; // esi
  int v13; // edi
  __int64 v14; // rax
  __int64 v15; // rcx
  char v16; // dl
  bool v17; // zf
  const char *v18; // rbx
  const char *v19; // rax
  const char *v21; // rax
  int v22; // [rsp+60h] [rbp+18h] BYREF
  __int64 v23; // [rsp+68h] [rbp+20h] BYREF

  v4 = love_version();
  v5 = 0i64;
  do
  {
    v6 = a115[v5++];
    if ( v6 != *(_BYTE *)(v4 + v5 - 1) )
    {
      v21 = (const char *)love_version();
      printf("Version mismatch detected!\nLOVE binary is version %s\nLOVE library is version %s\n", "11.5", v21);
      return 1i64;
    }
  }
  while ( v5 != 5 );
  while ( argc <= 1 || strcmp((const char *)argv[1], "--version") )
  {
    lua_State = (struct lua_State *)luaL_newstate();
    luaL_openlibs(lua_State);
    lua_getfield(lua_State, 0xFFFFD8EEi64, "package");
    lua_getfield(lua_State, 0xFFFFFFFFi64, "preload");
    lua_pushcclosure(lua_State, &luaopen_love_jitsetup, 0i64);
    lua_setfield(lua_State, 0xFFFFFFFEi64, "love.jitsetup");
    lua_settop(lua_State, 0xFFFFFFFDi64);
    lua_getfield(lua_State, 0xFFFFD8EEi64, "require");
    lua_pushstring(lua_State, "love.jitsetup");
    lua_call(lua_State, 1i64, 0i64);
    lua_getfield(lua_State, 0xFFFFD8EEi64, "package");
    lua_getfield(lua_State, 0xFFFFFFFFi64, "preload");
```

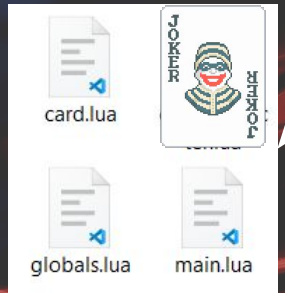
House always Wins

- Worked a little better

```
PS C:\Users\camps\Projects\Balatro> .\fakemain.exe
Hey Mom, I'm a DLL function being called!
Love Version: 11.5
Love Codename: Mysterious Mysteries
Console opened successfully
Ended execution
```

- Would have to do this so many times and some of the DLL imports are hard to understand how to use correctly
- Time/Skill issue

Beating the House



Beating the House

- LÖVE ships builds as a Self-Extracting Archive that contains the Lua source files
- This is why you can view the source files with 7zip

- We can abuse this
- “Extract” files, modify Lua source code, repack into SFX .exe

Beating the House

```
patcher = LuaPatcher(sys.argv[1])

patcher.patch_lines_in_file("main.lua", "G:draw()", [modded_draw_func])

patcher.patch_lines_in_file("main.lua", 129, [modded_global_vars])

patcher.patch_append_function(modded_calculate_score)

patcher.repack()
```

Cards

- Card is the general object filled out for each moveable card in the game and includes all their functionality
- Differentiates by “type”
- Joker, voucher, playing card, tarot, planet…
- Most information is stored in Game.lua regarding the attributes of the cards, and Card.lua calls for that information to make the card in the game environment.
-

Jokers

Lots of organizational attributes pertaining to type, image, cost, and more.

```
self.P_CENTERS = {
    c_base={max = 500, freq = 2, line = 'base', name = "Default Base", pos = {x=1,y=0}, set = "Default", label = 'Base Card', effect = "Base", cost_mult = 1.0, config = {}},

    --Jokers
    j_joker=          {order = 1,  unlocked = true,   start_alerted = true, discovered = true,  blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 2, name
    j_sophomore=      {order = 151,  unlocked = true, discovered = true, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 2, name = "Sophomore", pos = {x
    j_greedy_joker=   {order = 2,  unlocked = true,   discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 5, name = "Greedy Joker", pos
    j_lusty_joker=    {order = 3,  unlocked = true,   discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 5, name = "Lusty Joker", pos =
    j_wrathful_joker= {order = 4,  unlocked = true,   discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 5, name = "Wrathful Joker", pos
    j_gluttenous_joker= {order = 5,  unlocked = true,   discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 5, name = "Gluttonous Joker",
    j_jolly=          {order = 6,  unlocked = true,   discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 3, name = "Jolly Joker", pos =
    j_zany=           {order = 7,  unlocked = true,   discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 4, name = "Zany Joker", pos =
    j_mad=            {order = 8,  unlocked = true,   discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 4, name = "Mad Joker", pos = {
    j_crazy=          {order = 9,  unlocked = true,   discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 4, name = "Crazy Joker", pos =
    j_droll=          {order = 10,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 4, name = "Droll Joker", pos =
    j_sly=            {order = 11,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 3, name = "Sly Joker",set = "J
    j_wily=           {order = 12,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 4, name = "Wily Joker",set = "
    j_clever=         {order = 13,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 4, name = "Clever Joker",set =
    j_devious=        {order = 14,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 4, name = "Devious Joker",set
    j_crafty=         {order = 15,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 4, name = "Crafty Joker",set =

    j_half=           {order = 16,  unlocked = true,   discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 5, name = "Half Joker", pos =
    j_stencil=        {order = 17,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 2, cost = 8, name = "Joker Stencil", pos
    j_four_fingers=   {order = 18,  unlocked = true,  discovered = false, blueprint_compat = false, eternal_compat = true, rarity = 2, cost = 7, name = "Four Fingers", pos
    j_mime=           {order = 19,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 2, cost = 5, name = "Mime", pos = {x=4,y
    j_credit_card=    {order = 20,  unlocked = true,  discovered = false, blueprint_compat = false, eternal_compat = true, rarity = 1, cost = 1, name = "Credit Card", pos
    j_ceremonial=     {order = 21,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 2, cost = 6, name = "Ceremonial Dagger",
    j_banner=         {order = 22,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 5, name = "Banner", pos = {x=
    j_mystic_summit=  {order = 23,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 5, name = "Mystic Summit", pos
    j_marble=         {order = 24,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 2, cost = 6, name = "Marble Joker", pos
    j_loyalty_card=   {order = 25,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 2, cost = 5, name = "Loyalty Card", pos
    j_8_ball=         {order = 26,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 5, name = "8 Ball", pos = {x=0
    j_misprint=       {order = 27,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 4, name = "Misprint", pos = {x
    j_dusk=           {order = 28,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 2, cost = 5, name = "Dusk", pos = {x=4,y
    j_raised_fist=    {order = 29,  unlocked = true,  discovered = false, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 5, name = "Raised Fist", pos =
```

The ability is only the last part, the config attribute

```
, rarity = 1, cost = 2, name = "Joker", pos = {x=0,y=0}, set = "Joker", effect = "Mult", cost_mult = 1.0, config = {mult = 4}},
 name = "Sophomore", pos = {x=0,y=0}, set = "Joker", effect = "Type Mult", cost_mult = 1.0, config = {t_mult = 0.1, type = 'High Card'}},
, name = "Greedy Joker", pos = {x=6,y=1}, set = "Joker", effect = "Suit Mult", cost_mult = 1.0, config = {extra = {s_mult = 4, suit = 'Diamonds'}}},
, name = "Lusty Joker", pos = {x=7,y=1}, set = "Joker", effect = "Suit Mult", cost_mult = 1.0, config = {extra = {s_mult = 4, suit = 'Hearts'}}},
, name = "Wrathful Joker", pos = {x=8,y=1}, set = "Joker", effect = "Suit Mult", cost_mult = 1.0, config = {extra = {s_mult = 4, suit = 'Spades'}}},
, name = "Gluttonous Joker", pos = {x=9,y=1}, set = "Joker", effect = "Suit Mult", cost_mult = 1.0, config = {extra = {s_mult = 4, suit = 'Clubs'}}},
, name = "Jolly Joker", pos = {x=2,y=0}, set = "Joker", effect = "Type Mult", cost_mult = 1.0, config = {t_mult = 8, type = 'Pair'}},
, name = "Zany Joker", pos = {x=3,y=0}, set = "Joker", effect = "Type Mult", cost_mult = 1.0, config = {t_mult = 12, type = 'Three of a Kind'}},
, name = "Mad Joker", pos = {x=4,y=0}, set = "Joker", effect = "Type Mult", cost_mult = 1.0, config = {t_mult = 20, type = 'Four of a Kind'}},
, name = "Crazy Joker", pos = {x=5,y=0}, set = "Joker", effect = "Type Mult", cost_mult = 1.0, config = {t_mult = 12, type = 'Straight'}},
, name = "Droll Joker", pos = {x=6,y=0}, set = "Joker", effect = "Type Mult", cost_mult = 1.0, config = {t_mult = 10, type = 'Flush'}},
, name = "Sly Joker",set = "Joker", config = {t_chips = 50, type = 'Pair'}, pos = {x=0,y=14}},
, name = "Wily Joker",set = "Joker", config = {t_chips = 100, type = 'Three of a Kind'}, pos = {x=1,y=14}},
, name = "Clever Joker",set = "Joker", config = {t_chips = 150, type = 'Four of a Kind'}, pos = {x=2,y=14}},
, name = "Devious Joker",set = "Joker", config = {t_chips = 100, type = 'Straight'}, pos = {x=3,y=14}},
, name = "Crafty Joker",set = "Joker", config = {t_chips = 80, type = 'Flush'}, pos = {x=4,y=14}},

5, name = "Half Joker", pos = {x=7,y=0}, set = "Joker", effect = "Hand Size Mult", cost_mult = 1.0, config = {extra = {mult = 20, size = 3}}},
, name = "Joker Stencil", pos = {x=2,y=5}, set = "Joker", effect = "Hand Size Mult", cost_mult = 1.0, config = {}},
7, name = "Four Fingers", pos = {x=6,y=6}, set = "Joker", effect = "", config = {}},
, name = "Mime", pos = {x=4,y=1}, set = "Joker", effect = "Hand card double", cost_mult = 1.0, config = {extra = 1}},
1, name = "Credit Card", pos = {x=5,y=1}, set = "Joker", effect = "Credit", cost_mult = 1.0, config = {extra = 20}},
, name = "Ceremonial Dagger", pos = {x=5,y=5}, set = "Joker", effect = "", config = {mult = 0}},
, name = "Banner", pos = {x=1,y=2}, set = "Joker", effect = "Discard Chips", cost_mult = 1.0, config = {extra = 40}},
, name = "Mystic Summit", pos = {x=2,y=2}, set = "Joker", effect = "No Discard Mult", cost_mult = 1.0, config = {extra = {mult = 15, d_remaining = 0}}},
, name = "Marble Joker", pos = {x=3,y=2}, set = "Joker", effect = "Stone card hands", cost_mult = 1.0, config = {extra = 1}},
, name = "Loyalty Card", pos = {x=4,y=2}, set = "Joker", effect = "1 in 10 mult", cost_mult = 1.0, config = {extra = {Xmult = 4, every = 5, remaining = "5 remaining"}}},
, name = "8 Ball", pos = {x=0,y=5}, set = "Joker", effect = "Spawn Tarot", cost_mult = 1.0, config = {extra= 2}},
, name = "Misprint", pos = {x=6,y=2}, set = "Joker", effect = "Random Mult", cost_mult = 1.0, config = {extra = {max = 23, min = 0}}},
, name = "Dusk", pos = {x=4,y=7}, set = "Joker", effect = "", config = {extra = 1}, unlock_condition = {type = '', extra = '', hidden = true}},
, name = "Raised Fist", pos = {x=8,y=2}, set = "Joker", effect = "Socialized Mult", cost_mult = 1.0, config = {}},
```

Jokers

```
elseif self.ability.name == 'Half Joker' then loc_vars = {self.ability.extra.mult, self.ability.extra.size}
elseif self.ability.name == 'Fortune Teller' then loc_vars = {self.ability.extra, (G.GAME.consumeable_usage_total and G.GAME.consumeable_usage_
elseif self.ability.name == 'Steel Joker' then loc_vars = {self.ability.extra, 1 + self.ability.extra*(self.ability.steel_tally or 0)}
elseif self.ability.name == 'Chaos the Clown' then loc_vars = {self.ability.extra}
elseif self.ability.name == 'Space Joker' then loc_vars = {''..(G.GAME and G.GAME.probabilities.normal or 1), self.ability.extra}
elseif self.ability.name == 'Stone Joker' then loc_vars = {self.ability.extra, self.ability.extra*(self.ability.stone_tally or 0)}
elseif self.ability.name == 'Drunkard' then loc_vars = {self.ability.d_size}
elseif self.ability.name == 'Green Joker' then loc_vars = {self.ability.extra.hand_add, self.ability.extra.discard_sub, self.ability.mult}
elseif self.ability.name == 'Credit Card' then loc_vars = {self.ability.extra}
elseif self.ability.name == 'Greedy Joker' or self.ability.name == 'Lusty Joker' or
    self.ability.name == 'Wrathful Joker' or self.ability.name == 'Gluttonous Joker' then loc_vars = {self.ability.extra.s_mult, localize(self.a
elseif self.ability.name == 'Blue Joker' then loc_vars = {self.ability.extra, self.ability.extra*((G.deck and G.deck.cards) and #G.deck.cards o
elseif self.ability.name == 'Sixth Sense' then loc_vars = {}
elseif self.ability.name == 'Mime' then
elseif self.ability.name == 'Hack' then loc_vars = {self.ability.extra+1}
elseif self.ability.name == 'Pareidolia' then
elseif self.ability.name == 'Faceless Joker' then loc_vars = {self.ability.extra.dollars, self.ability.extra.faces}
elseif self.ability.name == 'Oops! All 6s' then
elseif self.ability.name == 'Juggler' then loc_vars = {self.ability.h_size}
elseif self.ability.name == 'Golden Joker' then loc_vars = {self.ability.extra}
elseif self.ability.name == 'Joker Stencil' then loc_vars = {self.ability.x_mult}
elseif self.ability.name == 'Four Fingers' then
elseif self.ability.name == 'Ceremonial Dagger' then loc_vars = {self.ability.mult}
elseif self.ability.name == 'Banner' then loc_vars = {self.ability.extra}
elseif self.ability.name == 'Misprint' then
    local r_mults = {}
    for i = self.ability.extra.min, self.ability.extra.max do
        r_mults[#r_mults+1] = tostring(i)
```

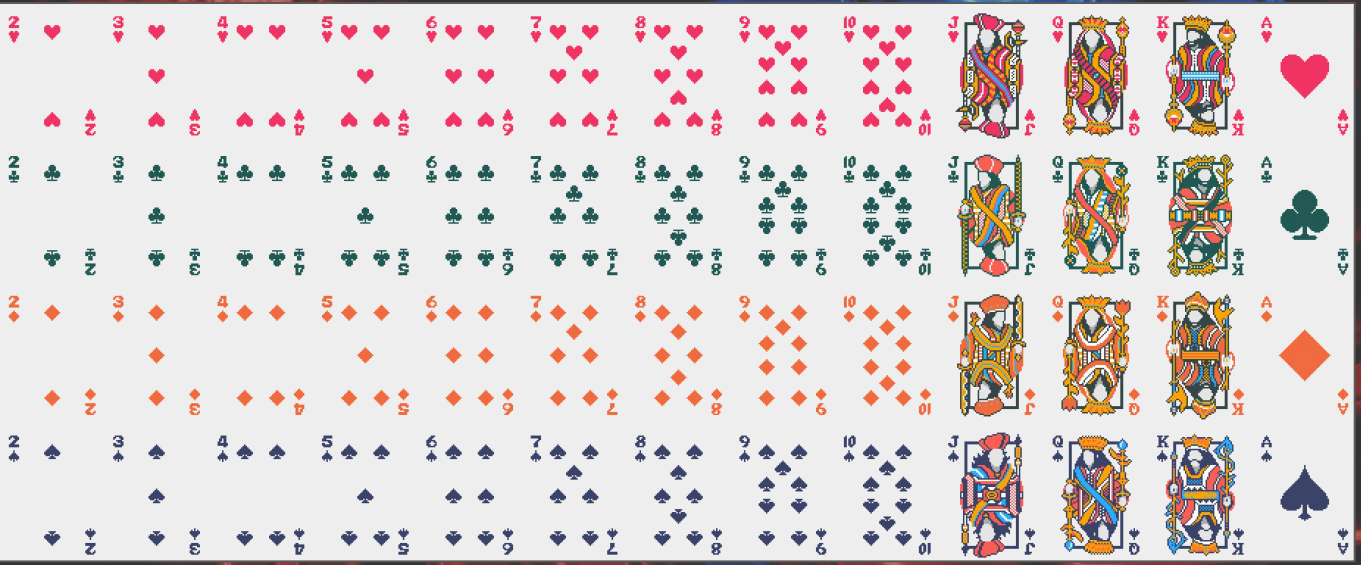
What a genius. This is in the Card class, where when initialized, it pulls the info from the Game class and assigns the ability to the Card object.

There are nearly 250 lines of if/else statements assigning text alone in the Card class

Jokers Cont.

The images from joker come from one big PNG file

This is the same for every card object. They're just recentered



New Hands

Five Fives- Balatro does not have much in the way of a central handler when it comes to game logic. Determining hand type, a major part of the game, is in misc_functions, while common_events.lua has logic for resetting a joker thats hardly used.

Creating a new hand involved creating the new hand in game.lua, ui descriptions in en_us.lua, and adding new logic to misc_functions.

Discovering the location of certain logics in the game proved tedious.

Challenges

The game has a challenge mode that can change any of the base rule

Since this game is a roguelike, I had to make a challenge to show all my changes at once

```
{
    name = 'LUKE TEST',
    id = 'c_luke_1',
    rules = {
        custom = {
            --{id = 'no_reward'},
        },
        modifiers = {
            {id = 'dollars', value = 4},
        }
    },
    jokers = {
        {id = 'j_sophomore'},

    },
    consumeables = {
    },
    vouchers = {
    },
    deck = {
        cards = {{s='D',r='5',},{s='D',r='5',},{s='D',r='5',},{
        type = 'Challenge Deck',
    },
    restrictions = {
        banned_cards = {
            {id = 'j_joker'},

        },
        banned_tags = {

        },
        banned_other = {

        }
    }
}
```

Coding Paradigms

LocalThunk, the developer, trounces many common coding paradigms.

- He uses hundreds of lines of if/else statements
- Places logic in random locations across files
- Moves code between updates for no reason
- Card and Card_area blur the lines on different classes, their code overlaps in functionality
- Redundant code is abundant, for instance: he includes many different ways to store the suit of a card yet doesn’t use all of them.
- He also creates an unnecessary number of different attributes, such as mult, s_mult, t_mult, rather than just applying mult.

The Sophomore VS The Joker

j_joker= {order = 1, unlocked = true, start_alerted = true, discovered = true, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 2, name = "Joker", pos = {x=0,y=0}, set = "Joker", effect = "Mult", cost_mult = 1.0, config = {mult = 4}},

j_sophomore= {order = 151, unlocked = true, discovered = true, blueprint_compat = true, eternal_compat = true, rarity = 1, cost = 2, name = "Sophomore", pos = {x=0,y=0}, set = "Joker", effect = "Type Mult", cost_mult = 1.0, config = {t_mult = 0.1, type = 'High Card'}},

Score Calculator

- Goal: Create a display that shows the score the current highlighted cards will receive when played
- This will require accessing the cards where they are stored and reading their properties
- Will also need to hijack the scoring method

Piercing the Veil

- There is a global singleton that manages pretty much everything in the game called “G”, which is defined in game.lua
- This is referenced for many functions and also contains things like hand info, card definitions, etc - truly a monolith

```lua
function Game:init()
    G = self

    self:set_globals()
end
```

=


```lua
self.cost = math.max(1, math.floor((self.base_cost + self.extra_cost + 0.5)*(100-G.GAME.discount_percent)/100))
if self.ability.set == 'Booster' and G.GAME.modifiers.booster_ante_scaling then self.cost = self.cost + G.GAME.round_resets.ante - 1 end
if self.ability.set == 'Booster' and (not G.SETTINGS.tutorial_complete) and G.SETTINGS.tutorial_progress and (not G.SETTINGS.tutorial_progress.completed_parts['shop_1']) then
```

Open Handed

- Cards contain bases with info like value (rank as string) and suit (also string)
  - Id and nominals are used for numerical values
- CardAreas contain a set of cards and keep track of which are selected, etc.
- G.hand is a CardArea containing all of the hand info
  - G.hand.cards has all the cards currently in hand
  - G.hand.highlighted contains selected cards
- So for instance we can call G.hand.highlighted[i].base.value for rank

Sorting the hand

- get_poker_hand_info and evaluate_poker_hand can be used to find the breakdown of a hand in terms of valid poker hands
- All possible hand are stored in a table
- We can just call this function to use it

```lua
G.FUNCS.get_poker_hand_info = function(_cards)
    local poker_hands = evaluate_poker_hand(_cards)
    local scoring_hand = {}
    local text,disp_text,loc_disp_text = 'NULL','NULL', 'NULL'
    if next(poker_hands["Flush Five"]) then text = "Flush Five"; scoring_hand = poker_hands["Flush Five"][1]
    elseif next(poker_hands["Flush House"]) then text = "Flush House"; scoring_hand = poker_hands["Flush House"][1]
    elseif next(poker_hands["Five of a Kind"]) then text = "Five of a Kind"; scoring_hand = poker_hands["Five of a Kind"][1]
    elseif next(poker_hands["Straight Flush"]) then text = "Straight Flush"; scoring_hand = poker_hands["Straight Flush"][1]
    elseif next(poker_hands["Four of a Kind"]) then text = "Four of a Kind"; scoring_hand = poker_hands["Four of a Kind"][1]
    elseif next(poker_hands["Full House"]) then text = "Full House"; scoring_hand = poker_hands["Full House"][1]
    elseif next(poker_hands["Flush"]) then text = "Flush"; scoring_hand = poker_hands["Flush"][1]
    elseif next(poker_hands["Straight"]) then text = "Straight"; scoring_hand = poker_hands["Straight"][1]
    elseif next(poker_hands["Three of a Kind"]) then text = "Three of a Kind"; scoring_hand = poker_hands["Three of a Kind"][1]
    elseif next(poker_hands["Two Pair"]) then text = "Two Pair"; scoring_hand = poker_hands["Two Pair"][1]
    elseif next(poker_hands["Pair"]) then text = "Pair"; scoring_hand = poker_hands["Pair"][1]
    elseif next(poker_hands["High Card"]) then text = "High Card"; scoring_hand = poker_hands["High Card"][1] end
```

```lua
function evaluate_poker_hand(hand)

  local results = {
    ["Flush Five"] = {},
    ["Flush House"] = {},
    ["Five of a Kind"] = {},
    ["Straight Flush"] = {},
    ["Four of a Kind"] = {},
    ["Full House"] = {},
    ["Flush"] = {},
    ["Straight"] = {},
    ["Three of a Kind"] = {},
    ["Two Pair"] = {},
    ["Pair"] = {},
    ["High Card"] = {},
    top = nil
  }

  local parts = {
    _5 = get_X_same(5,hand),
    _4 = get_X_same(4,hand),
    _3 = get_X_same(3,hand),
    _2 = get_X_same(2,hand),
    _flush = get_flush(hand),
    _straight = get_straight(hand),
    _highest = get_highest(hand)
  }
```

```lua
local text,disp_text,poker_hands,scoring_hand,non_loc_disp_text = G.FUNCS.get_poker_hand_info(G.hand.highlighted)
```

The World’s Biggest Calculator

- Over 500 lines and a seemingly infinite number of if statements are used to calculate the score of a hand in evaluate_play
- UI, card status, dollar amount, and more are also modified from this function

```lua
G.FUNCS.evaluate_play = function(e)
```

```lua
if not G.GAME.blind:debuff_hand(G.play.cards, poker_hands, text) then
    mult = mod_mult(G.GAME.hands[text].mult)
    hand_chips = mod_chips(G.GAME.hands[text].chips)
```

```lua
mult = mod_mult(G.GAME.hands[text].mult)
hand_chips = mod_chips(G.GAME.hands[text].chips)
```

```lua
mult = mod_mult(G.GAME.hands[text].mult)
hand_chips = mod_chips(G.GAME.hands[text].chips)
```

```lua
mult = mod_mult(G.GAME.hands[text].mult)
hand_chips = mod_chips(G.GAME.hands[text].chips)
```

```lua
--If chips added, do chip add event and add the chips to the total
if effects[ii].chips then
    if effects[ii].card then juice_card(effects[ii].card) end
    hand_chips = mod_chips(hand_chips + effects[ii].chips)
    update_hand_text({delay = 0}, {chips = hand_chips})
    card_eval_status_text(scoring_hand[i], 'chips', effects[ii].chips, percent)
end

--If mult added, do mult add event and add the mult to the total
if effects[ii].mult then
    if effects[ii].card then juice_card(effects[ii].card) end
    mult = mod_mult(mult + effects[ii].mult)
    update_hand_text({delay = 0}, {mult = mult})
    card_eval_status_text(scoring_hand[i], 'mult', effects[ii].mult, percent)
end

--If play dollars added, add dollars to total
if effects[ii].p_dollars then
    if effects[ii].card then juice_card(effects[ii].card) end
    ease_dollars(effects[ii].p_dollars)
    card_eval_status_text(scoring_hand[i], 'dollars', effects[ii].p_dollars, percent)
end

--If dollars added, add dollars to total
if effects[ii].dollars then
    if effects[ii].card then juice_card(effects[ii].card) end
    ease_dollars(effects[ii].dollars)
    card_eval_status_text(scoring_hand[i], 'dollars', effects[ii].dollars, percent)
end

--Any extra effects
```

MY Calculator

- ~200 lines and a seemingly less infinite number of if statements later, I can calculate the score without affecting anything else
- Pass the hand and it will return a score number without altering game state

```lua
calculate_score = function(highlightedCards, unhighlightedCards)
```

Unhighlighted cards must be passed because normally it removes the played cards from the hand

```lua
if not G.GAME.blind:debuff_hand(G.play.cards, poker_hands, text) then
    mult = mod_mult(G.GAME.hands[text].mult)
    hand_chips = mod_chips(G.GAME.hands[text].chips)
```

```lua
mult = mod_mult(G.GAME.hands[text].mult)
hand_chips = mod_chips(G.GAME.hands[text].chips)
```

```lua
mult = mod_mult(G.GAME.hands[text].mult)
hand_chips = mod_chips(G.GAME.hands[text].chips)
```

```lua
mult = mod_mult(G.GAME.hands[text].mult)
hand_chips = mod_chips(G.GAME.hands[text].chips)
```

```lua
--If chips added, do chip add event and add the chips to the total
if effects[ii].chips then
    [redacted]
    hand_chips = mod_chips(hand_chips + effects[ii].chips)
    [redacted]
end

--If mult added, do mult add event and add the mult to the total
if effects[ii].mult then
    [redacted]
    mult = mod_mult(mult + effects[ii].mult)
    [redacted]
end

--If play dollars added, add dollars to total
[redacted]

--Any extra effects
```

Where are these stored?

- Prints must be called from the love.draw function in main.lua, but everything else can be stored as a separate file as its own function

```lua
function love.draw()
    --Print info
    love.graphics.print(string.format("Current Hand: %s", highlighted_cards), 10, 10)
    love.graphics.print(string.format("Current Hand Type: %s", hand_type), 10, 30)
    love.graphics.print(string.format("Predicted Score: %s", calculated_score), 10, 50)
```

```lua
--Calculate current hand score
local calculated_score = 0
if G.hand then
    if G.hand.highlighted ~= nil then
        local highlightedCards = G.hand.highlighted
        local unhighlightedCards = {}
        for i=1, #G.hand.cards do
            if not G.hand.cards[i].highlighted then
                table.insert(unhighlightedCards, G.hand.cards[i])
            end
        end
        calculated_score = calculate_score(highlightedCards, unhighlightedCards)
    end
end
```

```lua
calculate_score = function(highlightedCards, unhighlightedCards)
```

Taking it Further

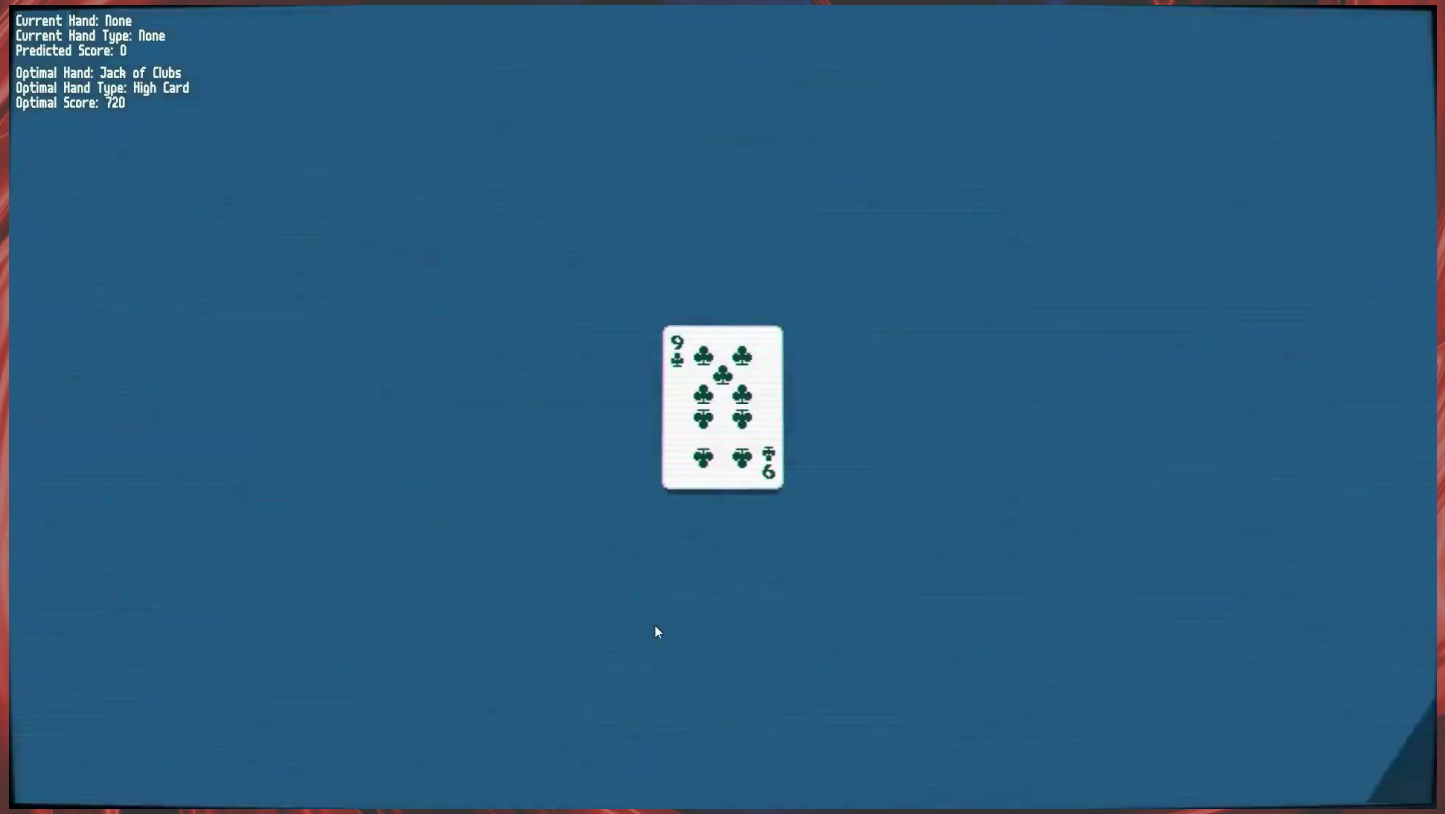
- What if we made the calculator check all possible hands to find the highest possible score?
- Can also have the hand automatically highlighted and played

```lua
function get_optimal_hand()
    local optimal_hand = {}
    local optimal_hand_string = "None"
    local optimal_hand_type = "None"
    local optimal_score = 0

    --Try all possible combinations of cards
    for c1 = 1, #G.hand.cards do
        optimal_hand, optimal_score = is_optimal_by_indexes({c1}, optimal_hand, optimal_score)
        for c2 = c1 + 1, #G.hand.cards do
            optimal_hand, optimal_score = is_optimal_by_indexes({c1, c2}, optimal_hand, optimal_score)
            for c3 = c2 + 1, #G.hand.cards do
                optimal_hand, optimal_score = is_optimal_by_indexes({c1, c2, c3}, optimal_hand, optimal_score)
                for c4 = c3 + 1, #G.hand.cards do
                    optimal_hand, optimal_score = is_optimal_by_indexes({c1, c2, c3, c4}, optimal_hand, optimal_score)
                    for c5 = c4 + 1, #G.hand.cards do
                        optimal_hand, optimal_score = is_optimal_by_indexes({c1, c2, c3, c4, c5}, optimal_hand, optimal_score)
                    end
                end
            end
        end
    end

    --Get hand string and hand type of optimal hand
    optimal_hand_string = card_table_to_string(optimal_hand)
    local text,disp_text,poker_hands,scoring_hand,non_loc_disp_text = G.FUNCS.get_poker_hand_info(optimal_hand)
    optimal_hand_type = disp_text

    return optimal_hand, optimal_hand_string, optimal_hand_type, optimal_score
end
```

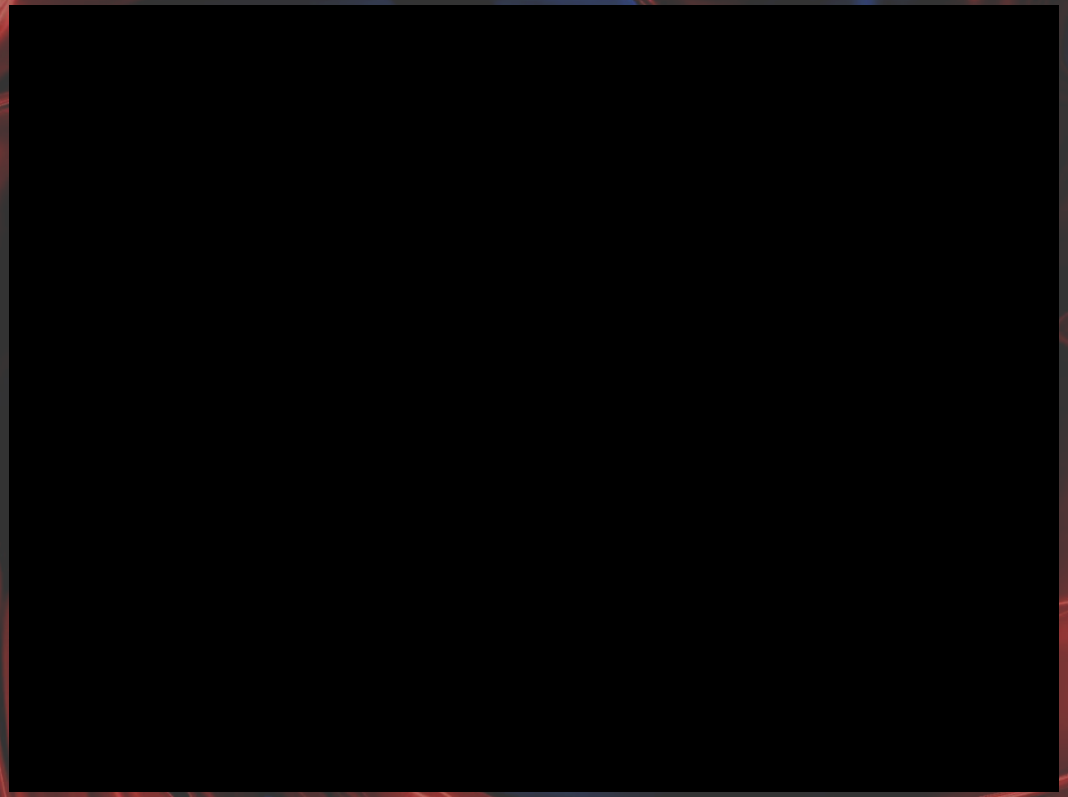
```lua
function force_hand_to(cards, silent)
    G.hand:unhighlight_all()
    for i = 1, #cards do
        G.hand:add_to_highlighted(cards[i], silent)
    end
end
```

```lua
G.FUNCS.play_cards_from_highlighted()
```

Score Calculator Demo

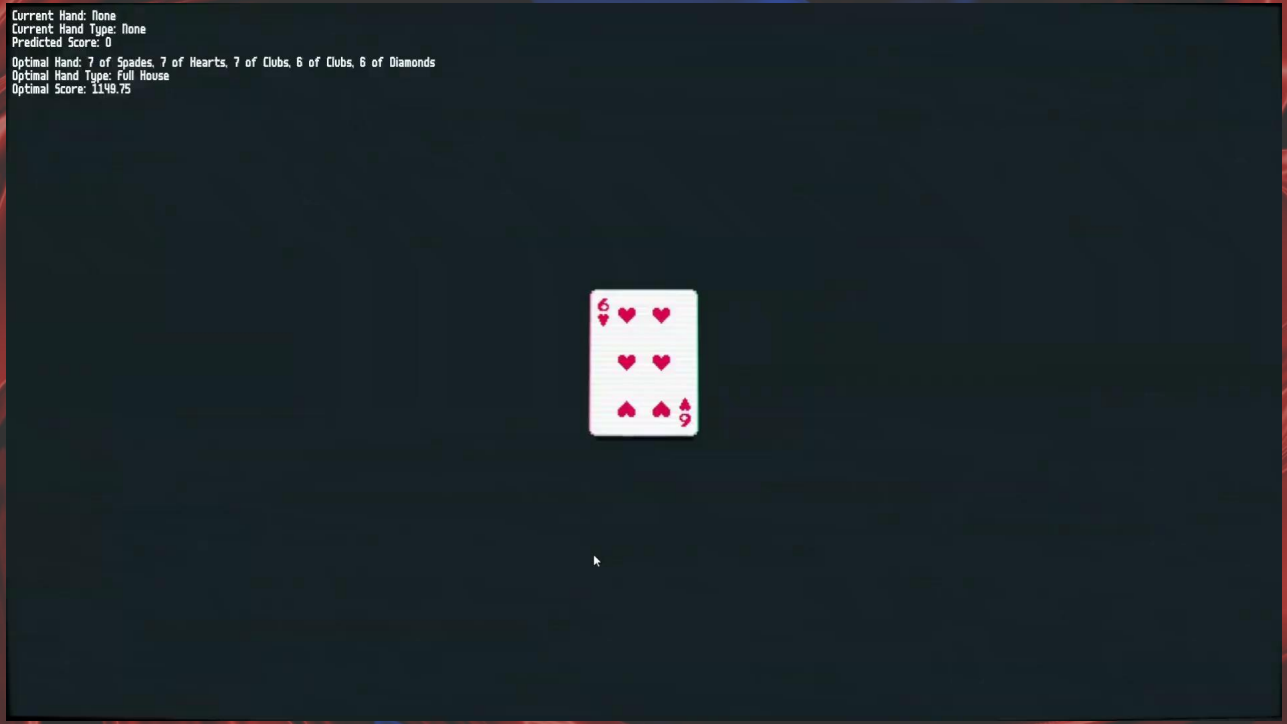


Added Keybinds



And now a silly one

- I had one more idea…



How I did it

- I was able to do this by calling various functions on the card class and by modifying its base values

```
function all_king_of_spades(cards)
    for i = 1, #cards do
        cards[i].base.value = 'King'
        cards[i].base.nominal = 10
        cards[i].base.face_nominal = 0.3
        cards[i].base.id = 13
        cards[i].base.suit = 'Spades'
        cards[i].base.suit_nominal = 0.04
        cards[i].base.suit_nominal_original = 0.004
        cards[i]:set_sprites(G.P_CENTERS.j_card_sharp, nil)
    end
end
```


Balatro
Amber Acorn
Flips and shuffles all Joker cards
Score at least
100,000
Reward: $$$$$$$$
Round score
238,592
0 X 0
Run Info
Hands
3
Discards
3
$51
Options
Ante
9 / 8
Round
26
0/0
JOKER
You Aced it!
YOU WIN!
Best Hand
364,044
Most Played Hand
Straight (25)
Cards Played
162
Ante
9
Cards Discarded
204
Round
26
Cards Purchased
96
Times Rerolled
41
New Run
New Discoveries
0
Main Menu
Seed
LQTS1JPG
Copy
Endless Mode
JUSTICE
2/2
56/56