Start.lua

```
return dofile( lua_path .. "Main_1.lua" )
```

Main_1.lua

```
local   GRID_SIZE = 3
local   GRID_MARGIN = 40.0
local   BLOCK_COUNT_MAX = ( GRID_SIZE * GRID_SIZE )
local   FONT_SIZE = 200.0
local   COLOR_ACTIVE = 2
local   COLOR_INACTIVE = 1
local   COLOR_FOCUS = 0
local   script_table = {}
local   array_number = {}
local   target_number = 1
local   last_time = -1.0

-- 初期化
function script_table:initialize()
    -- 番号配列を作成
    create_array_number()

    -- グリッドの初期状態を設定
    self.set_title_text( "Click¥nin order" )
    self.set_score_text( "Score:0" )
    self.grid_size = GRID_SIZE
    self.grid_margin = GRID_MARGIN

    for i=1, BLOCK_COUNT_MAX do
        local   info = {}
        info.is_active = true
        info.color = COLOR_ACTIVE
        info.font_size = FONT_SIZE
        info.text = array_number[i]
        self.block_info[i] = info
    end

    -- 経過時間をリセット
    elapsed_time = 0.0
end

-- 実行
function script_table:execute( delta_seconds )
    if( last_time <= 0.0 ) then
        self.set_score_text( string.format( "Time:%.2f", elapsed_time ) )
    end
end

-- クリック時
function script_table:on_click( index )
    if( ( self.block_info[ index ].is_active ) and
        ( self.block_info[ index ].text == target_number ) ) then
        self.block_info[ index ].is_active = false
        self.block_info[ index ].color = COLOR_INACTIVE
        self.update_grid( { index } )

        if( target_number < BLOCK_COUNT_MAX ) then
            target_number = target_number + 1
        else
            last_time = elapsed_time
            self.set_score_text( string.format( "Time:%.2f¥nCLEAR !!", last_time ) )
        end
    end
```

```lua
end

-- オーバーラップ時
function script_table:on_overlap( index, is_on )
    if( self.block_info[ index ].is_active ) then
        self.block_info[ index ].color = ( is_on and COLOR_FOCUS or COLOR_ACTIVE )
        self.update_grid( { index } )
    end
end

-- 番号配列を作成
function create_array_number()
    local   array_tmp = {}
    for i=1, BLOCK_COUNT_MAX do
        table.insert( array_tmp, i )
    end
    math.randomseed( os.time() )

    while( #array_tmp > 1 ) do
        local   index = math.random( 1, #array_tmp )
        table.insert( array_number, array_tmp[ index ] )
        table.remove( array_tmp, index )
    end
    table.insert( array_number, array_tmp[1] )
end

return script_table
```