Start.lua

```
return dofile( lua_path .. "Main_2.lua" )
```

Main_2.lua

```
local   GRID_SIZE = 4
local   GRID_MARGIN = 0.0
local   BLOCK_COUNT_MAX = ( GRID_SIZE * GRID_SIZE )
local   FONT_SIZE = 200.0
local   COLOR_LIGHT_ON = 3
local   COLOR_LIGHT_OFF = 1
local   COLOR_FOCUS = 0
local   script_table = {}
local   array_flag_active = {}
local   count_move = 0

-- 初期化
function script_table:initialize()
    -- アクティブフラグ配列を作成
    create_array_flag_active()

    -- グリッドの初期状態を設定
    self.set_title_text( "Turn off¥nall lights" )
    self.set_score_text( "Move:0" )
    self.grid_size = GRID_SIZE
    self.grid_margin = GRID_MARGIN

    for i=1, BLOCK_COUNT_MAX do
        local       info = {}
        info.is_active = array_flag_active[i]
        info.color = ( info.is_active and COLOR_LIGHT_ON or COLOR_LIGHT_OFF )
        info.font_size = FONT_SIZE
        info.text = ""
        self.block_info[i] = info
    end
end

-- 実行
function script_table:execute( delta_seconds )
end

-- クリック時
function script_table:on_click( index )
    if( self.block_info[ index ].is_active ) then
        -- 十字インデクス配列を取得
        local       array_index = get_array_index_cross( index )

        -- ブロックを反転
        for i=1, #array_index do
            reverse_block( self, array_index[i] )
        end

        -- グリッドを更新
        self.update_grid( array_index )

        -- スコアテキストを設定
        count_move = count_move + 1
        count_active = get_count_active( self )
        self.set_score_text( string.format( "Move:%d¥n%s", count_move, ( ( count_active < 1 ) and "CLEAR !!" or ""
 ) ) )
    end
end
```

```lua
-- 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_LIGHT_ON )
        self.update_grid( { index } )
    end
end

-- アクティブフラグ配列を作成
function create_array_flag_active()
    for i=1, BLOCK_COUNT_MAX do
        table.insert( array_flag_active, false )
    end
    math.randomseed( os.time() )

    local       count_reverse = ( GRID_SIZE * 3 ) + math.random( 1, GRID_SIZE )
    for i=1, count_reverse do
        local       index_center = get_index_center()
        if( index_center < 1 ) then
            break
        end

        local       array_index = get_array_index_cross( index_center )
        for i=1, #array_index do
            local       index = array_index[i]
            array_flag_active[ index ] = ( array_flag_active[ index ] == false )
        end
    end
end

-- 中心インデクスを取得
function get_index_center()
    local       index_center = math.random( 1, BLOCK_COUNT_MAX )
    for i=1, BLOCK_COUNT_MAX do
        if( array_flag_active[ index_center ] == false ) then
            return index_center
        end
        index_center = index_center + 1
        if( index_center > BLOCK_COUNT_MAX ) then
            index_center = 1
        end
    end
    return 0
end

-- 十字インデクス配列を取得
function get_array_index_cross( index_center )
    local       cx = math.fmod( ( index_center - 1 ), GRID_SIZE )
    local       cy = ( ( index_center - 1 - cx ) / GRID_SIZE )
    local       array_cross = {}
    if( cy > 0 ) then
        table.insert( array_cross, ( ( cy - 1 ) * GRID_SIZE + cx + 1 ) )
    end
    if( cy < ( GRID_SIZE - 1 ) ) then
        table.insert( array_cross, ( ( cy + 1 ) * GRID_SIZE + cx + 1 ) )
    end
    if( cx > 0 ) then
        table.insert( array_cross, ( cy * GRID_SIZE + ( cx - 1 ) + 1 ) )
    end
    if( cx < ( GRID_SIZE - 1 ) ) then
        table.insert( array_cross, ( cy * GRID_SIZE + ( cx + 1 ) + 1 ) )
    end
    table.insert( array_cross, index_center )
    return array_cross
end
```

```
-- ブロックを反転
function reverse_block( self, index )
    local       is_active = self.block_info[ index ].is_active
    self.block_info[ index ].is_active = ( is_active == false )
    self.block_info[ index ].color = ( is_active and COLOR_LIGHT_OFF or COLOR_LIGHT_ON )
end

-- アクティブ数を取得
function get_count_active( self )
    local       count = 0
    for i=1, BLOCK_COUNT_MAX do
        if( self.block_info[i].is_active ) then
            count = count + 1
        end
    end
    return count
end

return script_table
```