```
function pre_calculate_kernel(beta, dx)
  @radius = get_polar_radius_matrix(SIZE_X, SIZE_Y) * dx
  @Br = size(beta) * @radius
  @kernel_shell = beta[floor(@Br)] * kernel_core(@Br % 1)
  @kernel = @kernel_shell / sum(@kernel_shell)
  @kernel_FFT = FFT_2D(@kernel)
  return @kernel, @kernel_FFT
end

function run_automaton(@world, @kernel, @kernel_FFT, mu, sigma, dt)
  if size(@world) is small
    @potential = elementwise_convolution(@kernel, @world)
  else
    @world_FFT = FFT_2D(@world)
    @potential_FFT = elementwise_multiply(@kernel_FFT, @world_FFT)
    @potential = FFT_shift(real_part(inverse_FFT_2D(@potential_FFT)))
  end
  @growth = growth_mapping(@potential, mu, sigma)
  @new_world = clip(@world + dt * @growth, 0, 1)
  return @new_world, @growth, @potential
end

function simulation()
  R, T, mu, sigma, beta = get_parameters()
  dx = 1/R;  dt = 1/T;  time = 0
  @kernel, @kernel_FFT = pre_calculate_kernel(beta, dx)
  @world = get_initial_configuration(SIZE_X, SIZE_Y)
  repeat
    @world, @growth, @potential = run_automaton(@world,
        @kernel, @kernel_FFT, mu, sigma, dt)
    time = time + dt
    display(@world, @potential, @growth)
  end
end
```