```
In [1]:    include("../Sandpile.jl")
           include("../old/Sandpile64.jl")
           include("../old/SandpileAbstract.jl")
           using .Sandpile
           using .Sandpile64
           using .SandpileAbstract
           using BenchmarkTools
```

Need to use a smaller grid and less grains for benchmarking

```
In [2]:    x, y = 50, 50; # Dimensions of grid
           N = 1000; # Number of grains to be added
           f_χ = 4; # Critical value for sandpile model
```

Benchmarking

```
In [3]:    @benchmark run_sandpileAbstract!(sandpile_initAbstract(x, y, "zero"), N)
```

```
Out[3]:    BenchmarkTools.Trial: 943 samples with 1 evaluation.
            Range (min … max):  3.513 ms … 20.371 ms   GC (min … max):  0.00% … 71.45%
            Time   (median):     3.958 ms              GC (median):     0.00%
            Time   (mean ± σ):   5.286 ms ±  3.346 ms  GC (mean ± σ):  20.89% ± 21.84%
```

```
            3.51 ms          Histogram: log(frequency) by time        17.4 ms <

            Memory estimate: 4.67 MiB, allocs estimate: 7007.
```

```
In [4]:    @benchmark run_sandpile64!(sandpile_init64(x, y, "zero"), N)
```

```
Out[4]:    BenchmarkTools.Trial: 1130 samples with 1 evaluation.
            Range (min … max):  2.927 ms … 17.400 ms   GC (min … max):  0.00% … 73.21%
            Time   (median):     3.291 ms              GC (median):     0.00%
            Time   (mean ± σ):   4.416 ms ±  3.137 ms  GC (mean ± σ):  23.98% ± 23.03%
```

```
            2.93 ms          Histogram: log(frequency) by time        16.1 ms <

            Memory estimate: 4.67 MiB, allocs estimate: 7007.
```

```
In [5]:    @benchmark run_sandpile!(sandpile_init(x, y, "zero"), N)
```

```
Out[5]:    BenchmarkTools.Trial: 1155 samples with 1 evaluation.
            Range (min … max):  2.918 ms … 18.718 ms   GC (min … max):  0.00% … 65.86%
            Time   (median):     3.252 ms              GC (median):     0.00%
            Time   (mean ± σ):   4.316 ms ±  2.974 ms  GC (mean ± σ):  23.30% ± 22.84%
```

```
            2.92 ms          Histogram: log(frequency) by time        15.1 ms <

            Memory estimate: 4.65 MiB, allocs estimate: 7006.
```

```
In [6]:    @benchmark run_sandpileAbstract!(sandpile_initAbstract(x, y, "random"), N)
```

```
            BenchmarkTools.Trial: 227 samples with 1 evaluation.
```

```
Out[6]:    Range (min … max):  13.960 ms … 40.087 ms   GC (min … max):  0.00% … 40.37%
           Time  (median):     20.301 ms              GC (median):     0.00%
           Time  (mean ± σ):   22.128 ms ±  4.949 ms  GC (mean ± σ):  12.93% ± 14.97%
```



```
       14 ms              Histogram: frequency by time            33.5 ms <

       Memory estimate: 12.62 MiB, allocs estimate: 20144.
```

In [7]:

```
@benchmark run_sandpile64!(sandpile_init64(x, y, "random"), N)
```

```
Out[7]:  BenchmarkTools.Trial: 221 samples with 1 evaluation.
           Range (min … max):  13.419 ms … 45.078 ms   GC (min … max):  0.00% … 22.21%
           Time  (median):     21.189 ms              GC (median):     0.00%
           Time  (mean ± σ):   22.686 ms ±  5.815 ms  GC (mean ± σ):  12.78% ± 14.67%
```



```
       13.4 ms             Histogram: frequency by time           42.6 ms <

       Memory estimate: 11.98 MiB, allocs estimate: 18997.
```

```
In [8]:  @benchmark run_sandpile!(sandpile_init(x, y, "random"), N)
```

```
Out[8]:  BenchmarkTools.Trial: 211 samples with 1 evaluation.
           Range (min … max):  14.676 ms … 54.271 ms   GC (min … max):  0.00% … 30.42%
           Time  (median):     21.054 ms              GC (median):     0.00%
           Time  (mean ± σ):   23.628 ms ±  6.454 ms  GC (mean ± σ):  13.45% ± 15.47%
```



```
       14.7 ms             Histogram: frequency by time           44.2 ms <

       Memory estimate: 12.35 MiB, allocs estimate: 19661.
```

Type Stability of Functions

```
In [9]:  @code_warntype sandpile_init(x, y, "zero")
```

```
MethodInstance for Main.Sandpile.sandpile_init(::Int64, ::Int64, ::String)
  from sandpile_init(x::Int64, y::Int64, setup::String) in Main.Sandpile at /Users/raarome
ro/sandpile-julia/code/Sandpile.jl:17
Arguments
  #self#::Core.Const(Main.Sandpile.sandpile_init)
  x::Int64
  y::Int64
  setup::String
Body::Matrix{UInt8}
1 ─ %1  = (setup == "random")::Bool
└──       goto #3 if not %1
2 ─ %3  = Core.apply_type(Main.Sandpile.Array, Main.Sandpile.UInt8)::Core.Const(Array{UInt
8})
│   %4  = (0:3)::Core.Const(0:3)
│   %5  = Main.Sandpile.rand(%4, x, y)::Matrix{Int64}
│   %6  = Main.Sandpile.convert(%3, %5)::Matrix{UInt8}
└──       return %6
3 ─ %8  = (setup == "zero")::Bool
└──       goto #5 if not %8
4 ─ %10 = Main.Sandpile.zeros(Main.Sandpile.UInt8, x, y)::Matrix{UInt8}
└──       return %10
5 ─       Main.Sandpile.error("`setup` must have value of `random` or `zero`")
```

```
                    └── Core.Const(:(return %12))
```

In [10]:
```julia
@code_warntype add_grain!(sandpile_init(x, y, "zero"))
```

```
MethodInstance for Main.Sandpile.add_grain!(::Matrix{UInt8})
  from add_grain!(z::Matrix{UInt8}) in Main.Sandpile at /Users/raaromero/sandpile-julia/co
de/Sandpile.jl:38
Arguments
  #self#::Core.Const(Main.Sandpile.add_grain!)
  z::Matrix{UInt8}
Body::Int64
1 ─ %1  = Main.Sandpile.size(z)::Tuple{Int64, Int64}
│   %2  = Base.getindex(%1, 1)::Int64
│   %3  = (1:%2)::Core.PartialStruct(UnitRange{Int64}, Any[Core.Const(1), Int64])
│   %4  = Main.Sandpile.rand(%3)::Int64
│   %5  = Main.Sandpile.size(z)::Tuple{Int64, Int64}
│   %6  = Base.getindex(%5, 2)::Int64
│   %7  = (1:%6)::Core.PartialStruct(UnitRange{Int64}, Any[Core.Const(1), Int64])
│   %8  = Main.Sandpile.rand(%7)::Int64
│   %9  = Base.getindex(z, %4, %8)::UInt8
│   %10 = (%9 + 1)::Int64
│         Base.setindex!(z, %10, %4, %8)
└──       return %10
```

In [11]:
```julia
@code_warntype is_unstable(sandpile_init(x, y, "zero"))
```

```
MethodInstance for Main.Sandpile.is_unstable(::Matrix{UInt8})
  from is_unstable(z::Matrix{UInt8}) in Main.Sandpile at /Users/raaromero/sandpile-julia/c
ode/Sandpile.jl:48
Arguments
  #self#::Core.Const(Main.Sandpile.is_unstable)
  z::Matrix{UInt8}
Body::Bool
1 ─ %1 = (#self#)(z, 4)::Bool
└──      return %1
```

In [12]:
```julia
@code_warntype avalanche!(sandpile_init(x, y, "zero"))
```

```
MethodInstance for Main.Sandpile.avalanche!(::Matrix{UInt8})
  from avalanche!(z::Matrix{UInt8}) in Main.Sandpile at /Users/raaromero/sandpile-julia/co
de/Sandpile.jl:68
Arguments
  #self#::Core.Const(Main.Sandpile.avalanche!)
  z::Matrix{UInt8}
Body::UInt8
1 ─ %1 = (#self#)(z, 4)::UInt8
└──      return %1
```

In [13]:
```julia
@code_warntype run_sandpile!(sandpile_init(x, y, "zero"), 100_000)
```

```
MethodInstance for Main.Sandpile.run_sandpile!(::Matrix{UInt8}, ::Int64)
  from run_sandpile!(z::Matrix{UInt8}, N::Int64; N_crit, f_χ) in Main.Sandpile at /Users/ra
aromero/sandpile-julia/code/Sandpile.jl:102
Arguments
  #self#::Core.Const(Main.Sandpile.run_sandpile!)
  z::Matrix{UInt8}
  N::Int64
Body::Vector{Int64}
1 ─ %1 = Main.Sandpile.:(var"#run_sandpile!#1")(false, 4, #self#, z, N)::Vector{Int64}
```

```
        └──           return %1
```