High-resolution fluid effects

Rasmussen et al, Smoke Simulation for Large Scale Phenomena (SIGGRAPH 2003)

High-resolution fluid effects

Nielsen et al, Out-Of-Core and Compressed Level Set Methods (TOG 2007)

High-resolution fluid effects

Horvath & Geiger, Directable, high-resolution simulation of fire on the GPU (SIGGRAPH 2009)

Introduction

So, what's holding us back?

- Curse of dimensionality : 8x cost to double the linear resolution (typically more, if the time step needs to be shrunk, too)

Introduction

So, what's holding us back?

- Curse of dimensionality : 8x cost to double the linear resolution (typically more, if the time step needs to be shrunk, too)
- Footprint : Simulations can't fit on a single system's RAM (out-of-core processing, cluster computing are possible remedies)

Introduction

So, what's holding us back?

- Curse of dimensionality : 8x cost to double the linear resolution (typically more, if the time step needs to be shrunk, too)
- Footprint : Simulations can't fit on a single system's RAM (out-of-core processing, cluster computing are possible remedies)
- Poor scalability : Kernels with worse-than-linear complexity dominate

Introduction

Cost and scalability of fluids simulation components :

- Most components : linear complexity & admit efficient parallelization
  e.g. velocity/particle update, levelset reinitialization, extrapolation, etc.

Introduction

Cost and scalability of fluids simulation components :

- Most components : linear complexity & admit efficient parallelization
  e.g. velocity/particle update, levelset reinitialization, extrapolation, etc.

- Poisson pressure projection becomes the bottleneck at high resolution

  - Objective : Solve a discrete Poisson equation $\Delta \mathbf{x} = \mathbf{f}$

  - Common solver in graphics : Preconditioned conjugate gradient

Introduction

Cost and scalability of fluids simulation components :

- Most components : linear complexity & admit efficient parallelization
  e.g. velocity/particle update, levelset reinitialization, extrapolation, etc.
- Poisson pressure projection becomes the bottleneck at high resolution
  - Objective : Solve a discrete Poisson equation $\Delta \mathbf{x} = \mathbf{f}$
  - Common solver in graphics : Preconditioned conjugate gradient
  - Popular Poisson preconditioner : Incomplete Cholesky Factorization
    - Introduced in graphics : *Foster & Fedkiw, "Practical animation of liquids", 2001*
    - Difficult to parallelize without compromising preconditioning efficiency
    - Still too many iterations required for high resolution simulations

Introduction

Poisson solvers with even more favorable complexity?

- "Fast Poisson Solvers" : *Cyclic reduction, FFT, FACR, etc.*
    - Extremely fast, when applicable
    - Not appropriate for irregularly shaped domains

Introduction

Poisson solvers with even more favorable complexity?

- "Fast Poisson Solvers" : *Cyclic reduction, FFT, FACR, etc.*
    - Extremely fast, when applicable
    - Not appropriate for irregularly shaped domains
- Multigrid methods
    - Potentially O(N) asymptotic complexity
    - Good parallel potential

Introduction

Poisson solvers with even more favorable complexity?

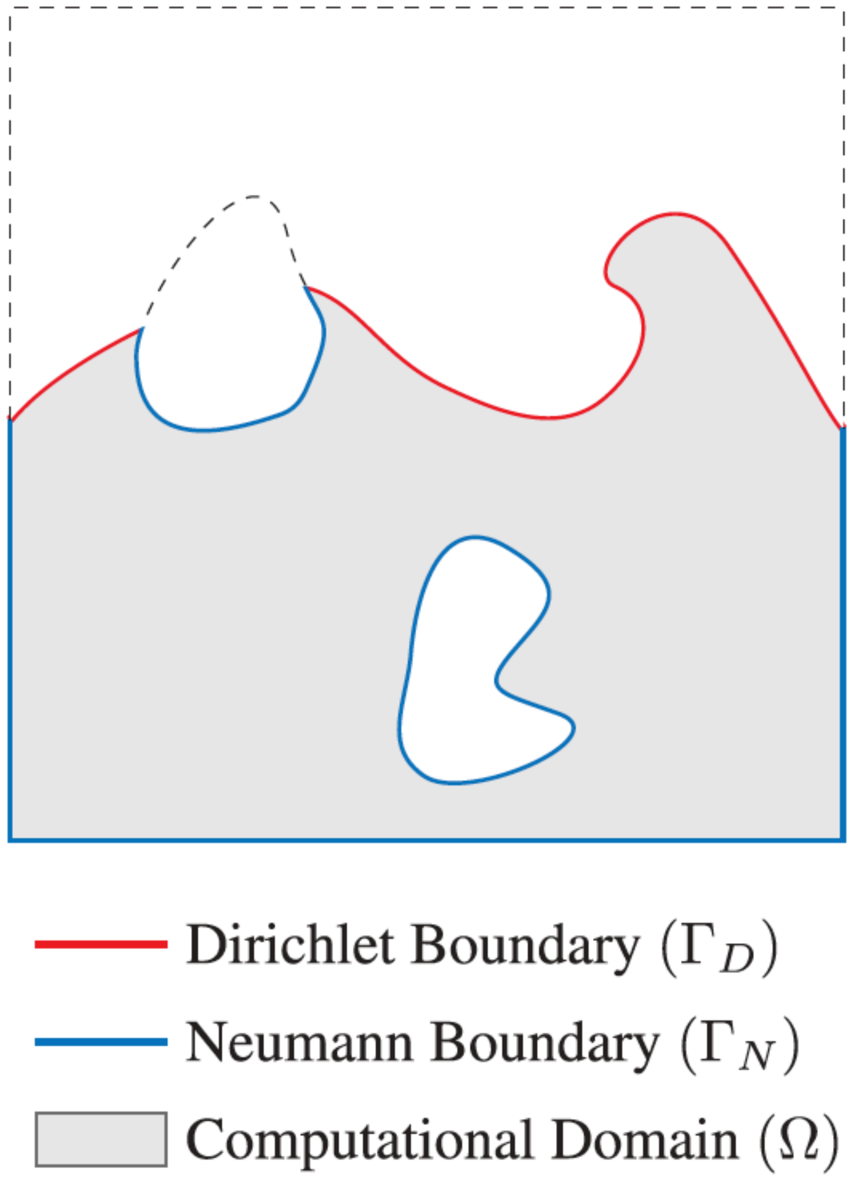
- "Fast Poisson Solvers" : *Cyclic reduction, FFT, FACR, etc.*
    - Extremely fast, when applicable
    - Not appropriate for irregularly shaped domains
- Multigrid methods
    - Potentially O(N) asymptotic complexity
    - Good parallel potential
    - Complications can compromise convergence performance
        - Irregular domain shapes, highly variable boundary conditions
        - Elaborate topological features (bubbles, fingers, slits, etc)

Problem description

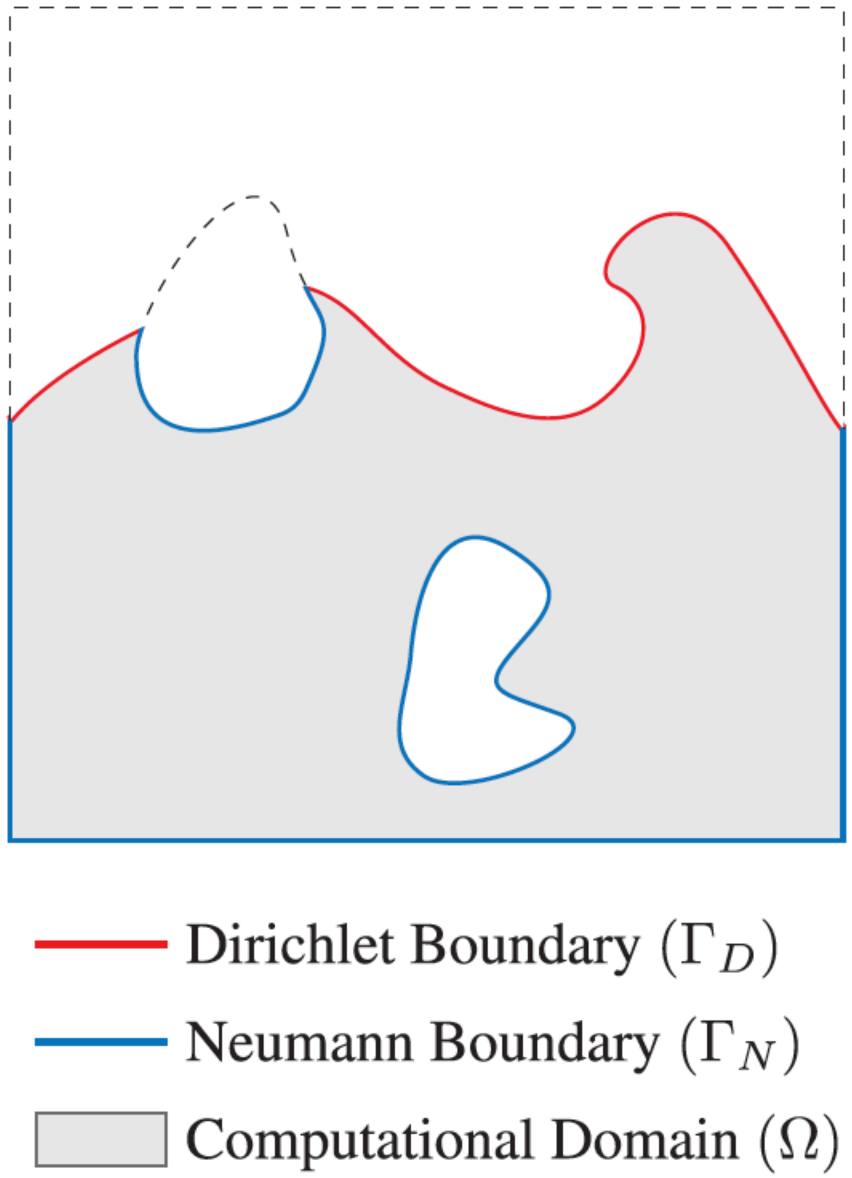
The Pressure Poisson equation

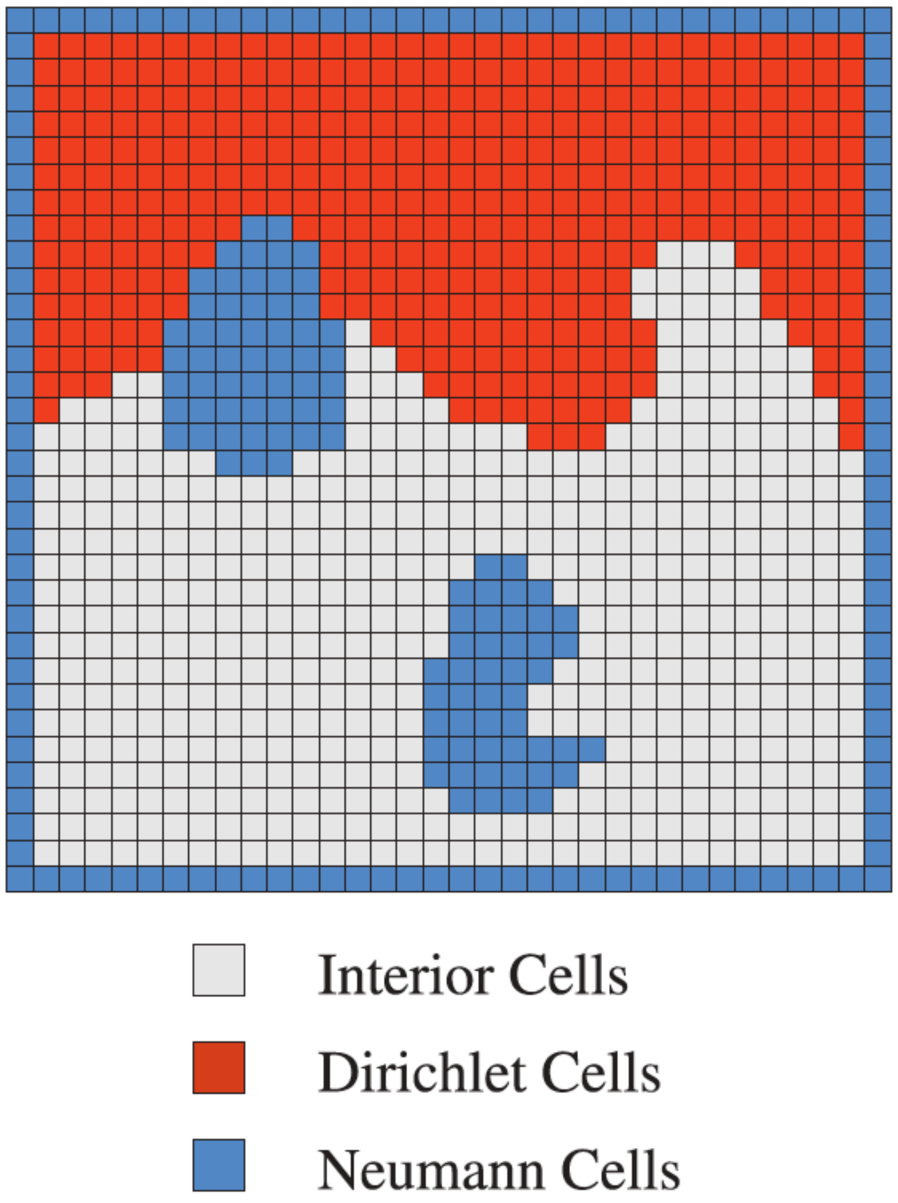
(for projecting a velocity field to its divergence-free component)

$$\Delta p = f \qquad \text{in } \Omega$$
$$p(\mathbf{x}) = \alpha(\mathbf{x}) \qquad \text{on } \Gamma_D$$
$$p_n(\mathbf{x}) = \beta(\mathbf{x}) \qquad \text{on } \Gamma_N$$



Problem description



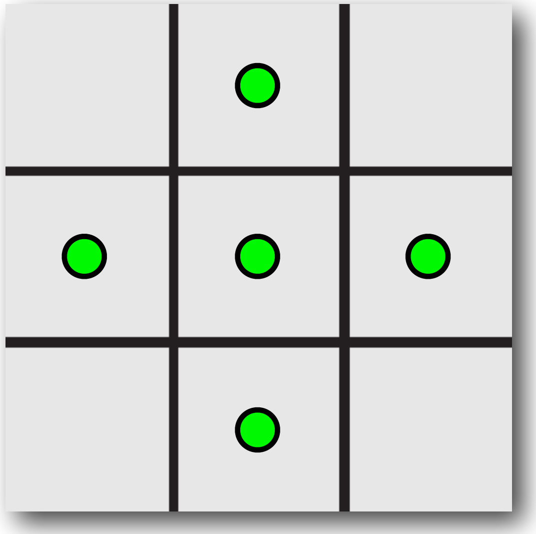


"Voxelized" Poisson problem

Problem description

Interior point discretization

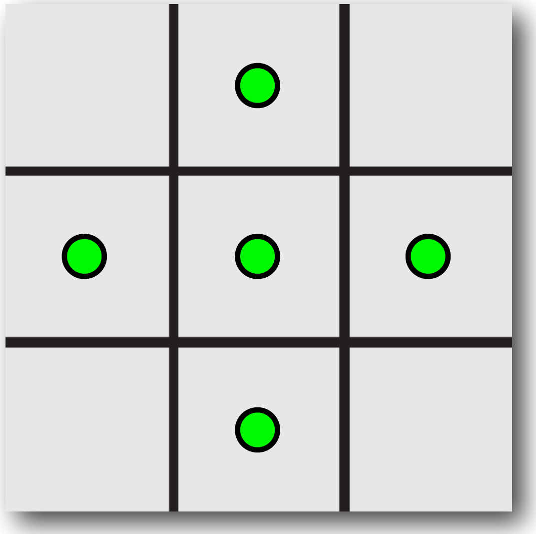
$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



Problem description

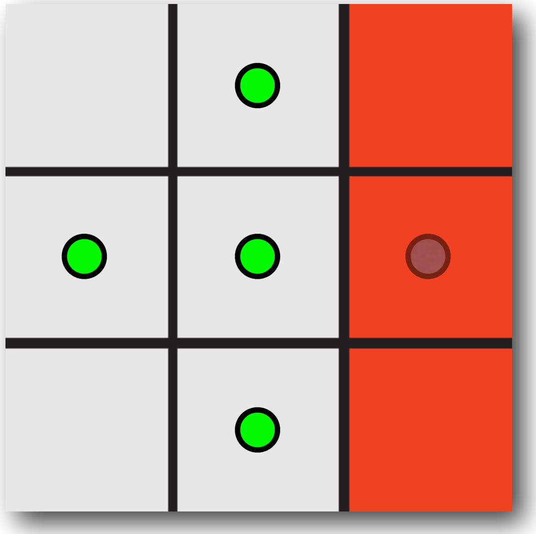
Interior point discretization

$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



Discretization near a Dirichlet boundary

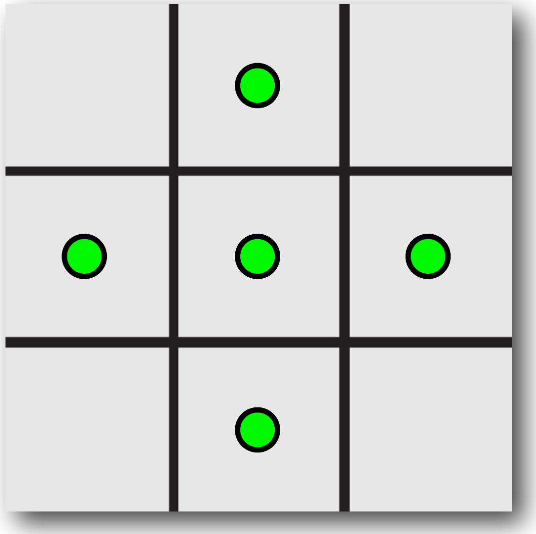
$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



Problem description

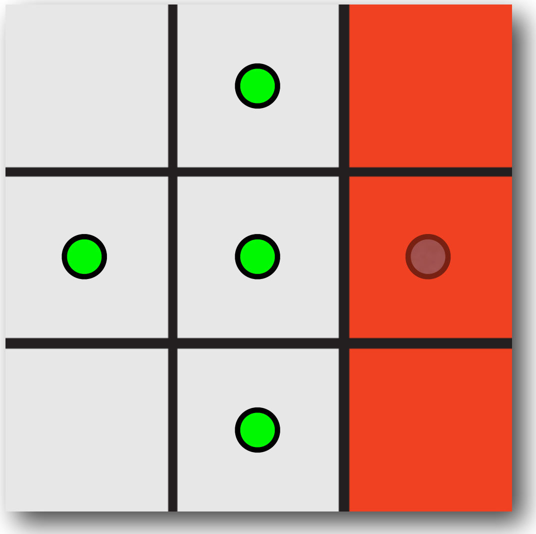
Interior point discretization

$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



Discretization near a Dirichlet boundary

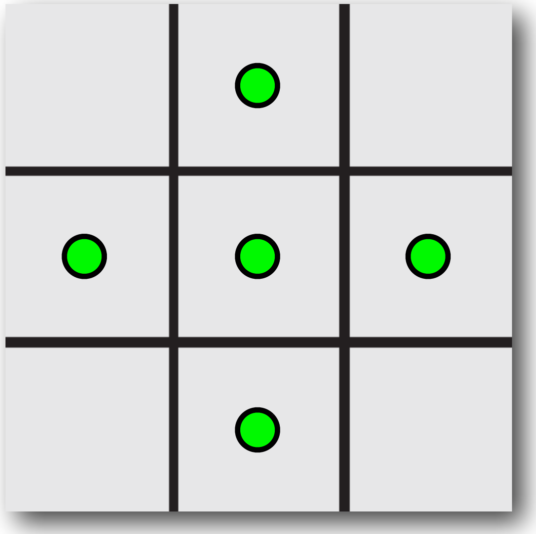
$$\frac{-4u_{ij} {\color{red} + u_{i+1,j}} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



Problem description

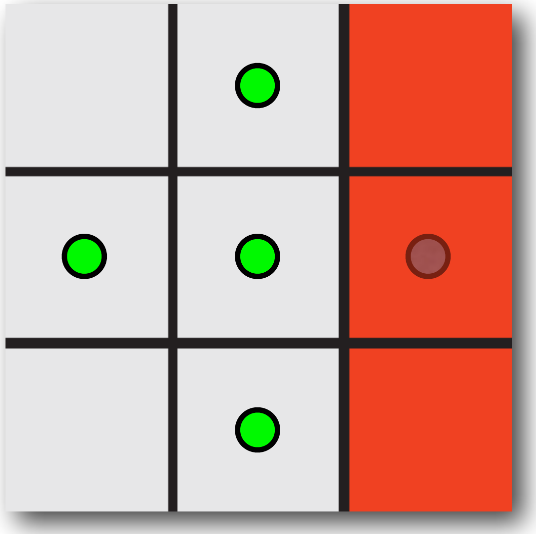
Interior point discretization

$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



Discretization near a Dirichlet boundary

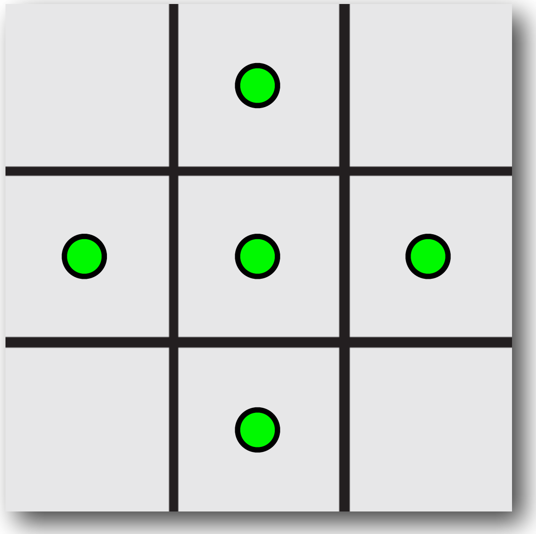
$$\frac{-4u_{ij} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij} - \frac{u_{i+1,j}}{h^2}$$



Problem description

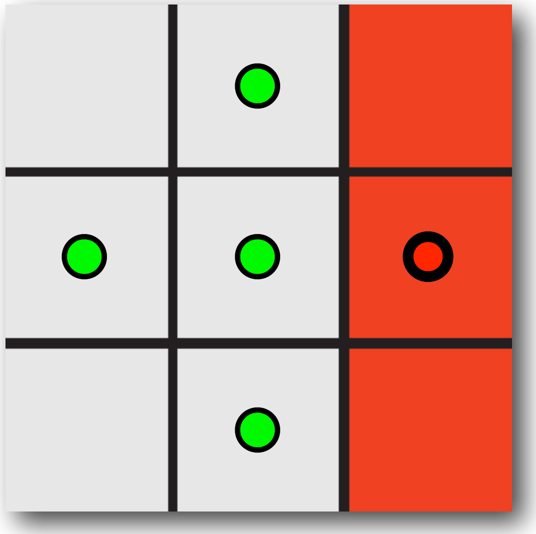
Interior point discretization

$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



Discretization near a Dirichlet boundary

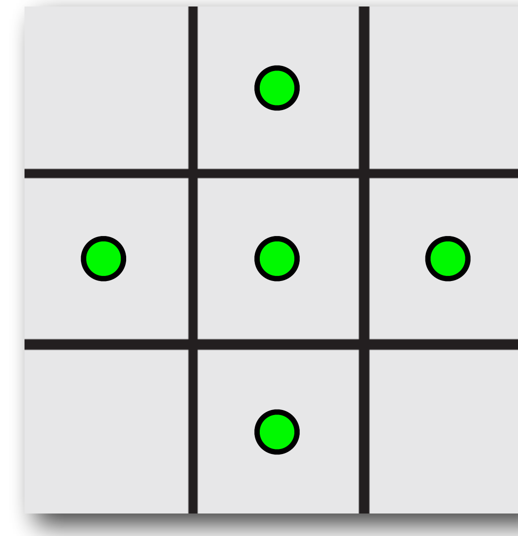
$$\frac{-4u_{ij} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij} - \frac{\alpha_{i+1,j}}{h^2}$$



Problem description

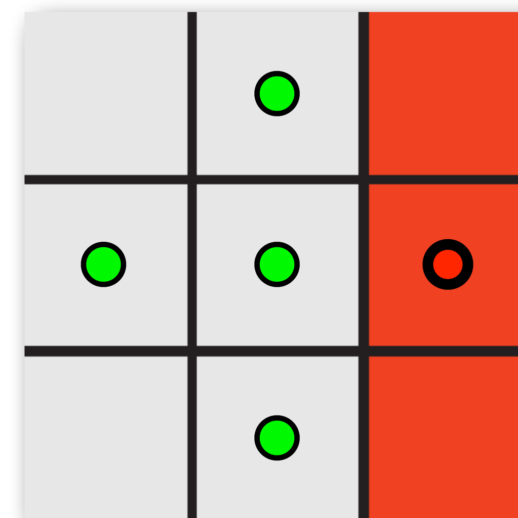
Interior point discretization

$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



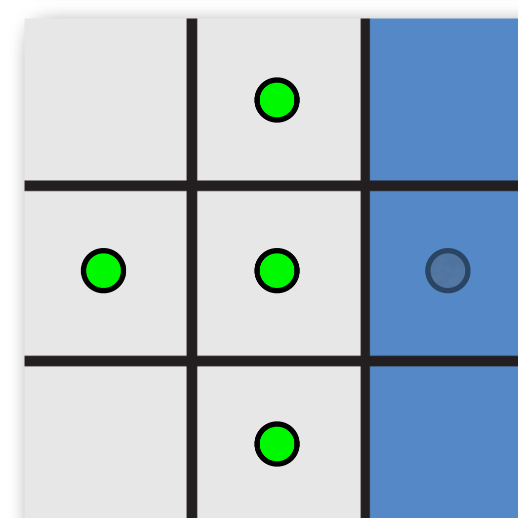
Discretization near a Dirichlet boundary

$$\frac{-4u_{ij} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij} - \frac{\alpha_{i+1,j}}{h^2}$$



Discretization near a Neumann boundary

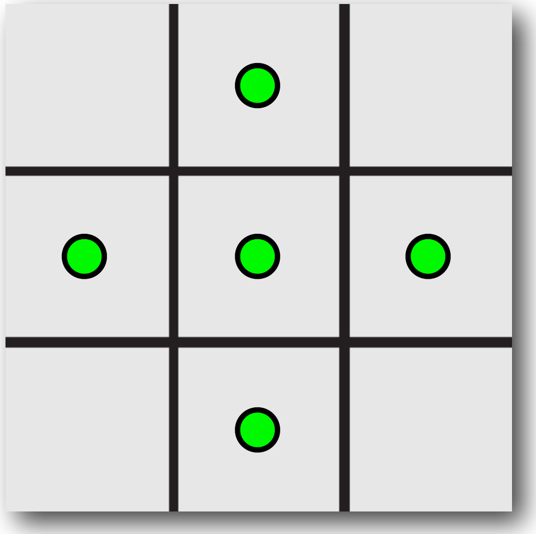
$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



Problem description

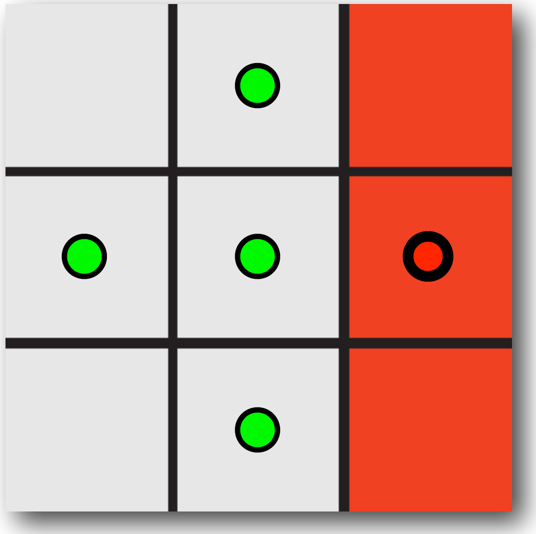
Interior point discretization

$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



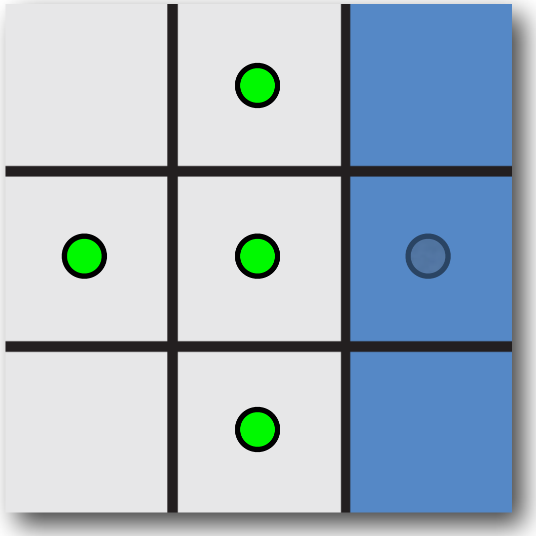
Discretization near a Dirichlet boundary

$$\frac{-4u_{ij} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij} - \frac{\alpha_{i+1,j}}{h^2}$$



Discretization near a Neumann boundary

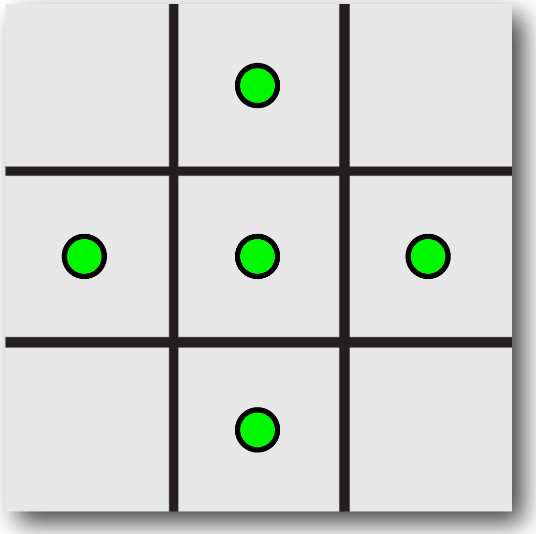
$$\frac{-3u_{ij} - u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



Problem description

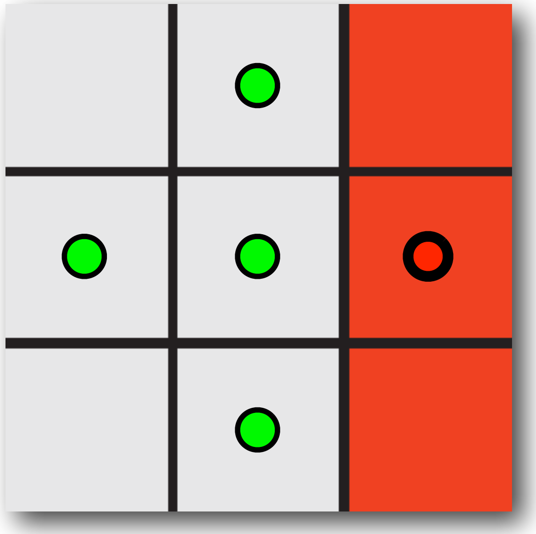
Interior point discretization

$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



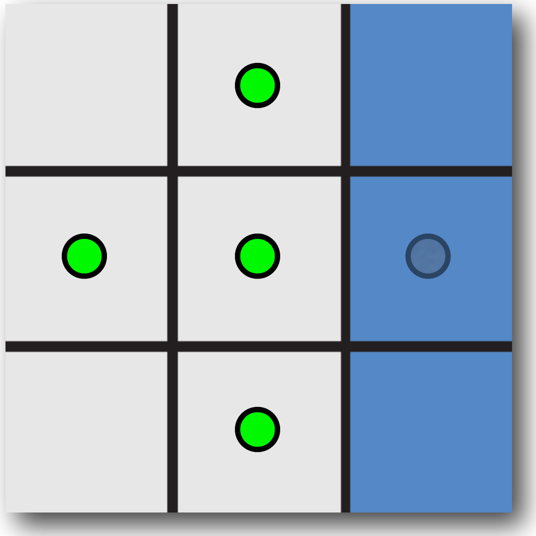
Discretization near a Dirichlet boundary

$$\frac{-4u_{ij} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij} - \frac{\alpha_{i+1,j}}{h^2}$$



Discretization near a Neumann boundary

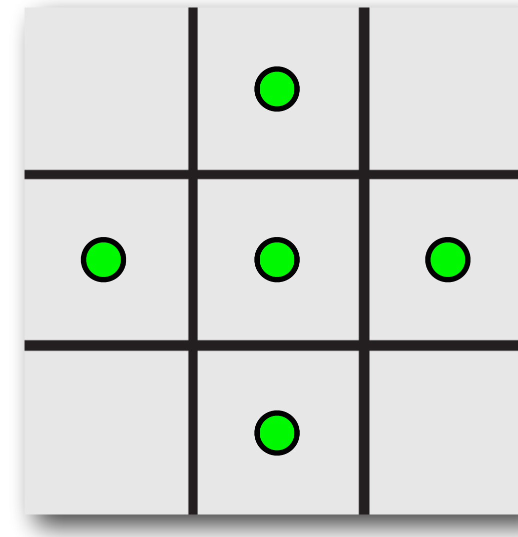
$$\frac{-3u_{ij} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij} - \frac{u_{i+1,j} - u_{ij}}{h^2}$$



Problem description

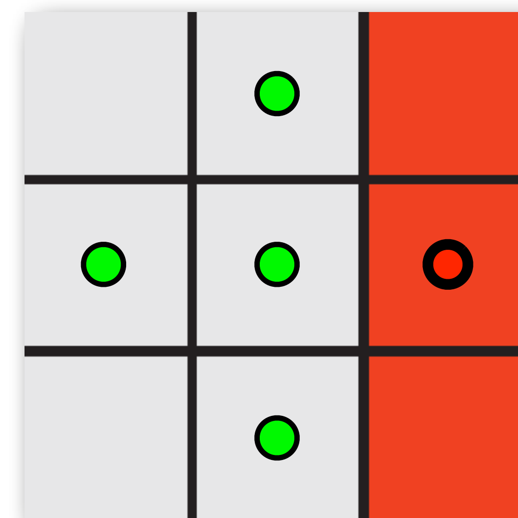
Interior point discretization

$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



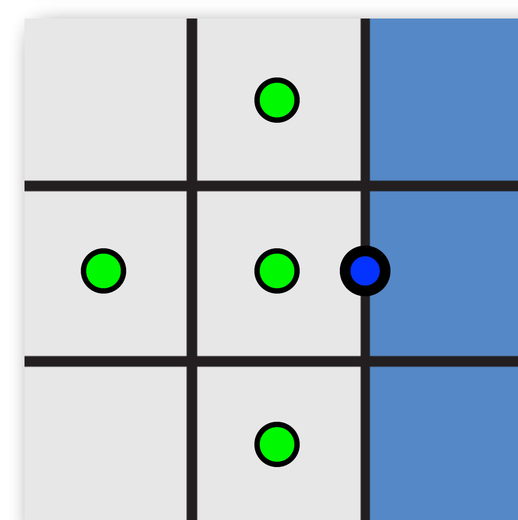
Discretization near a Dirichlet boundary

$$\frac{-4u_{ij} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij} - \frac{\alpha_{i+1,j}}{h^2}$$



Discretization near a Neumann boundary

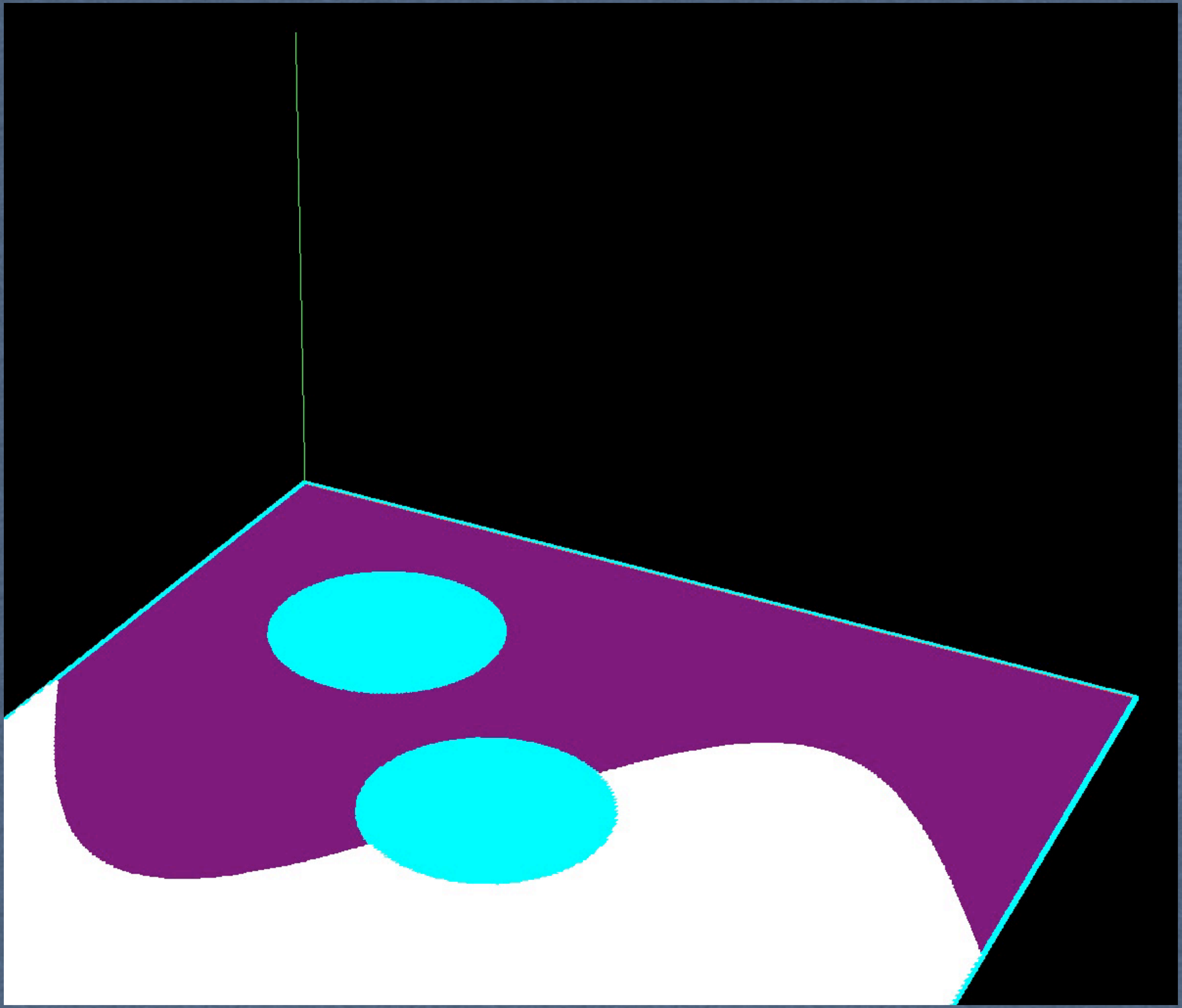
$$\frac{-3u_{ij} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij} - \frac{\beta_{i+\frac{1}{2},j}}{h}$$



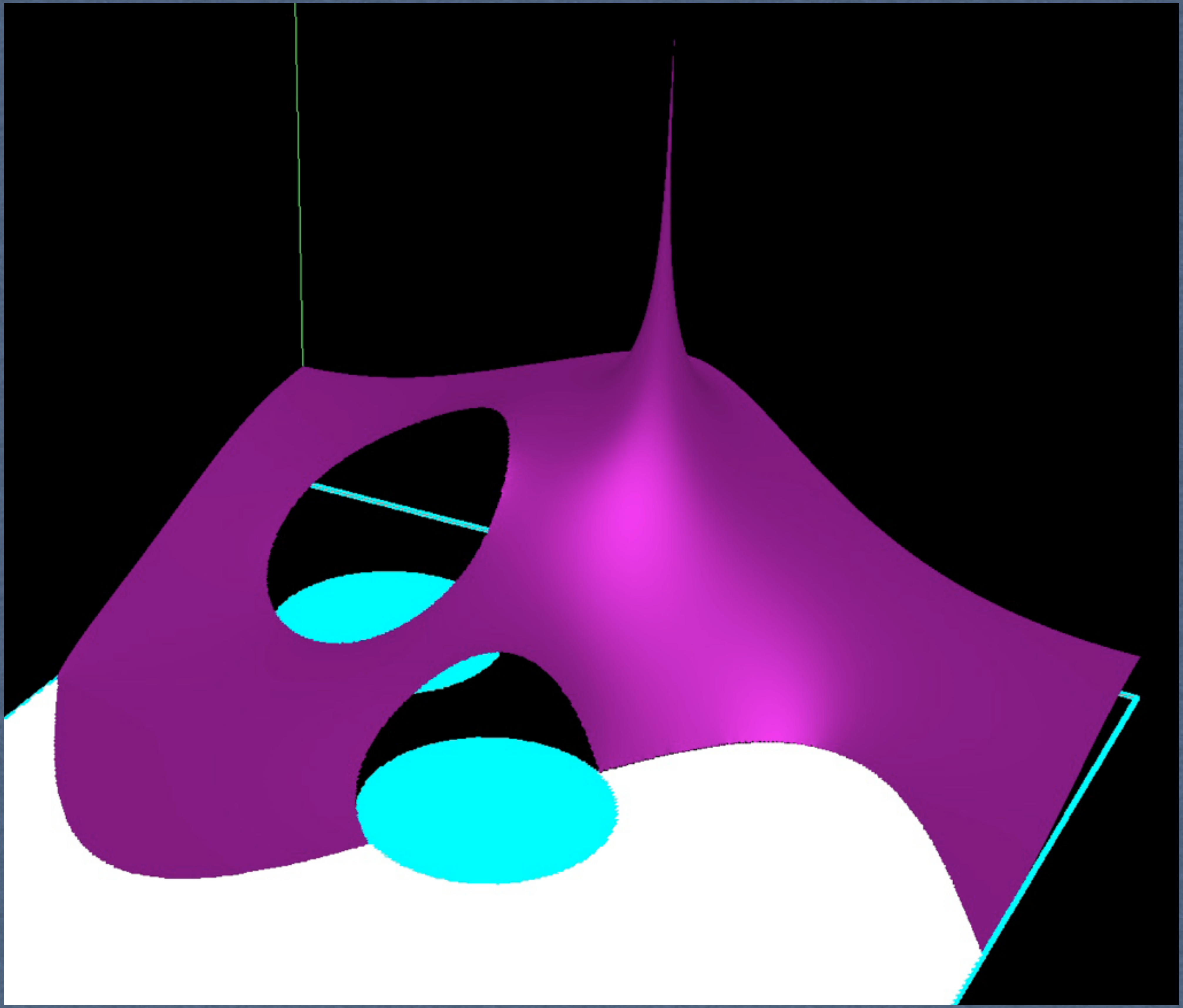
Problem description

Properties of the resulting discretization

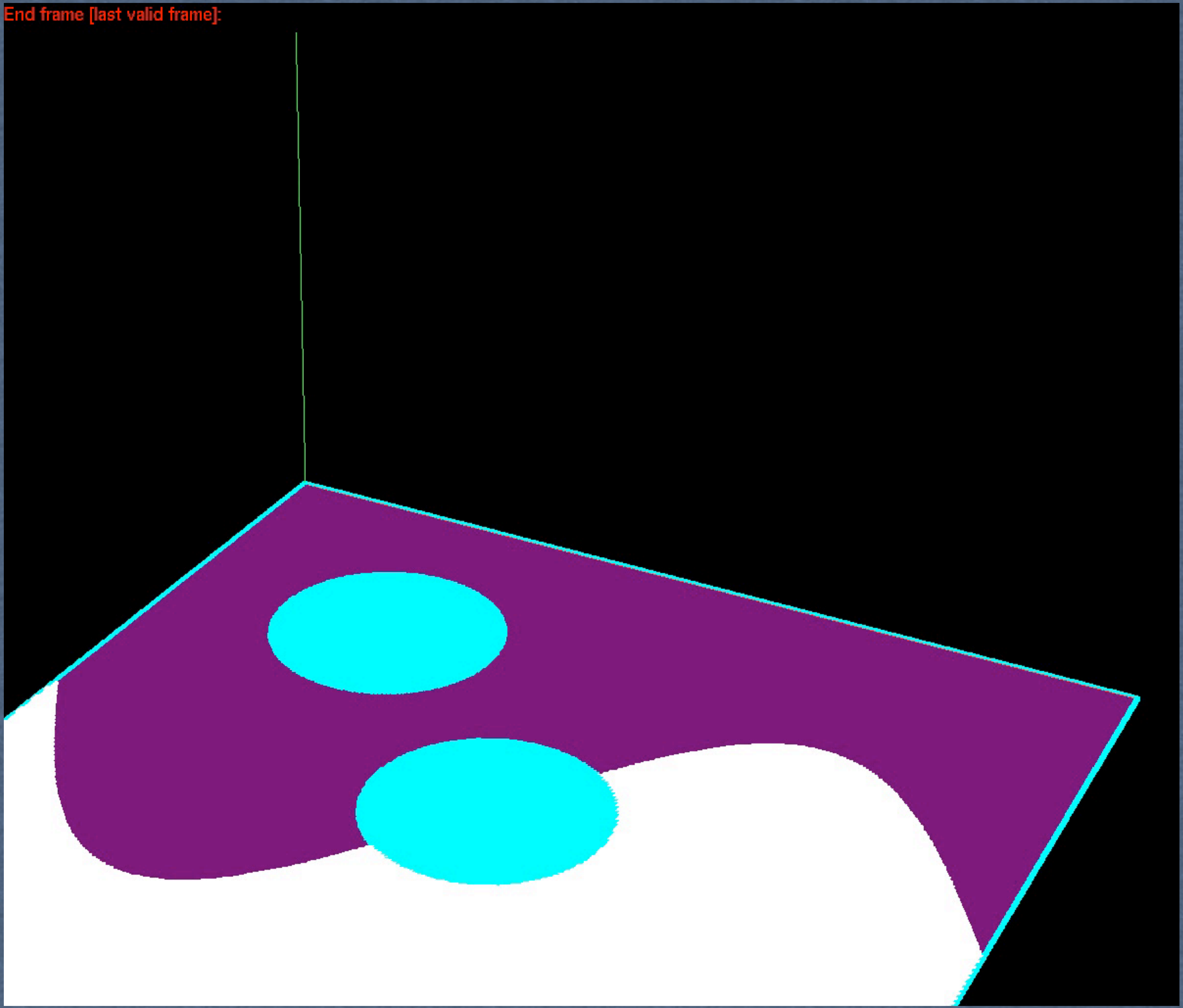
- Symmetric, sparse (banded) system
- Negative semi-definite (strictly definite with any Dirichlet boundary)
- Symmetric Krylov solvers are applicable, i.e. preconditioned CG



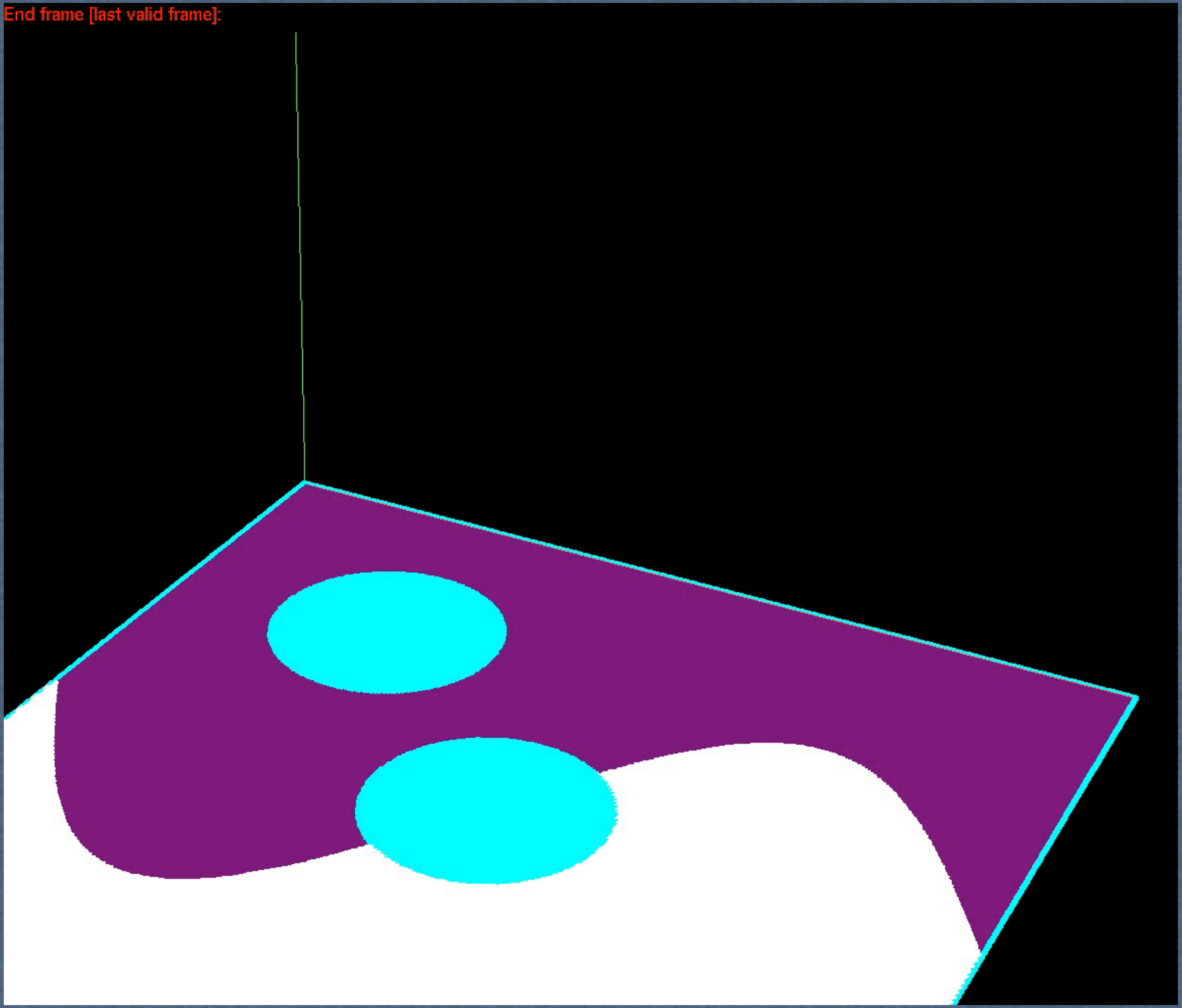
Sample 2D domain (512x512 resolution)



Exact solution



Conjugate Gradients (w/o preconditioning)



Conjugate Gradients (with a stock preconditioner)

Problem description

Properties of the resulting discretization

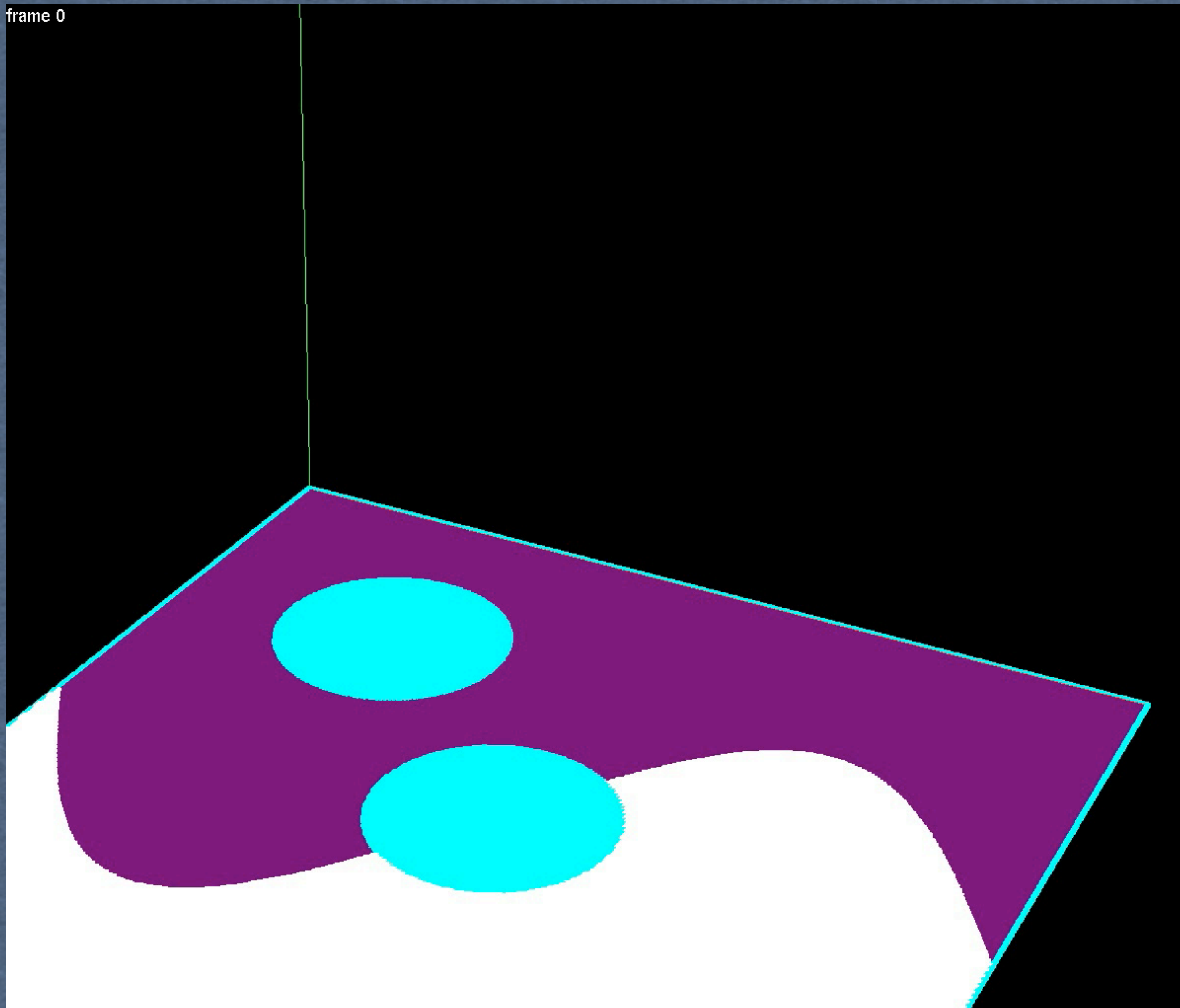
- Symmetric, sparse (banded) system
- Negative semi-definite (strictly definite with any Dirichlet boundary)
- Symmetric Krylov solvers are applicable, i.e. preconditioned CG

- *Convergence deterioration is even more pronounced in 3 dimensions, at higher resolutions and with more elaborate domain geometries*

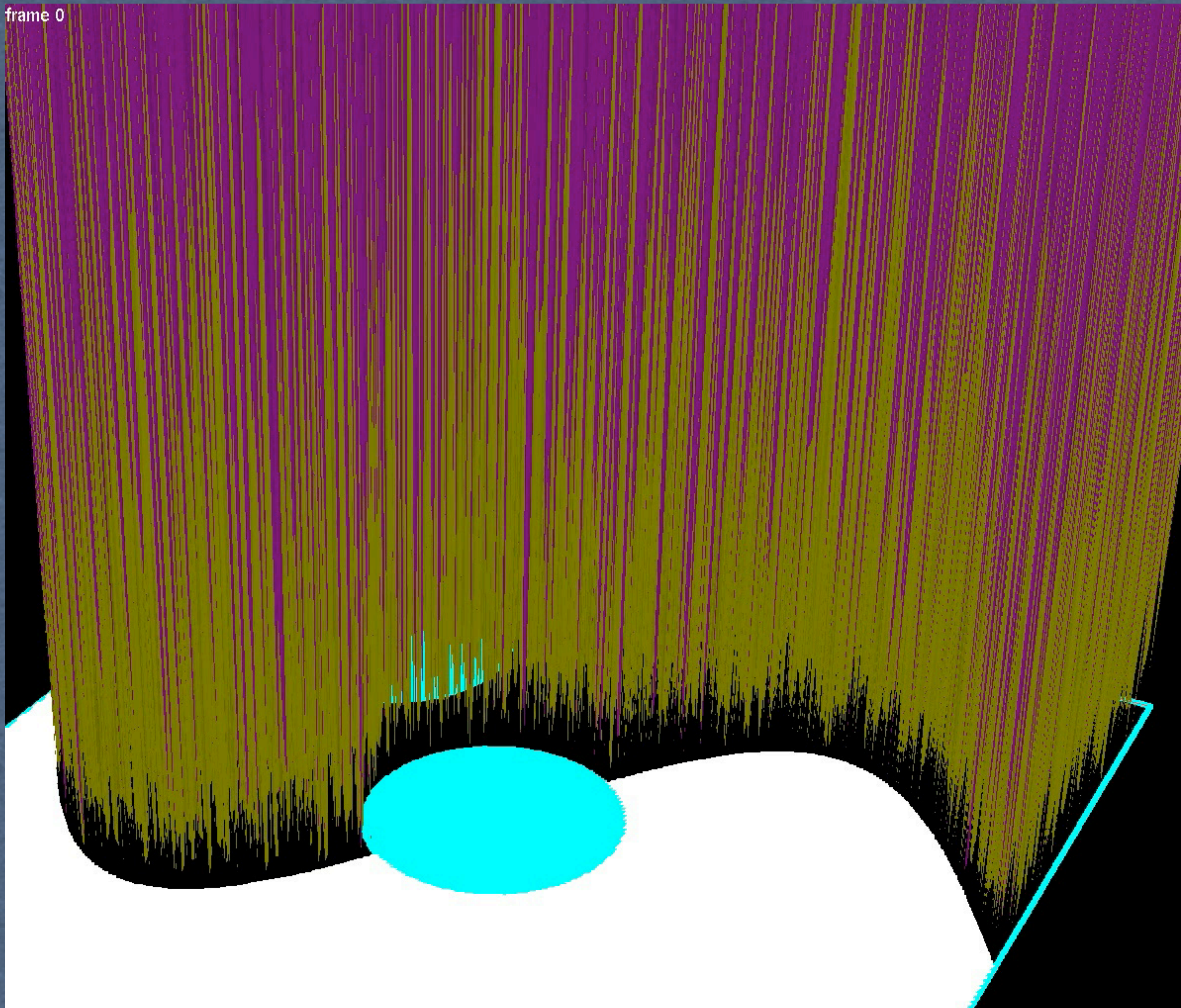
Multigrid

What about multigrid solvers?

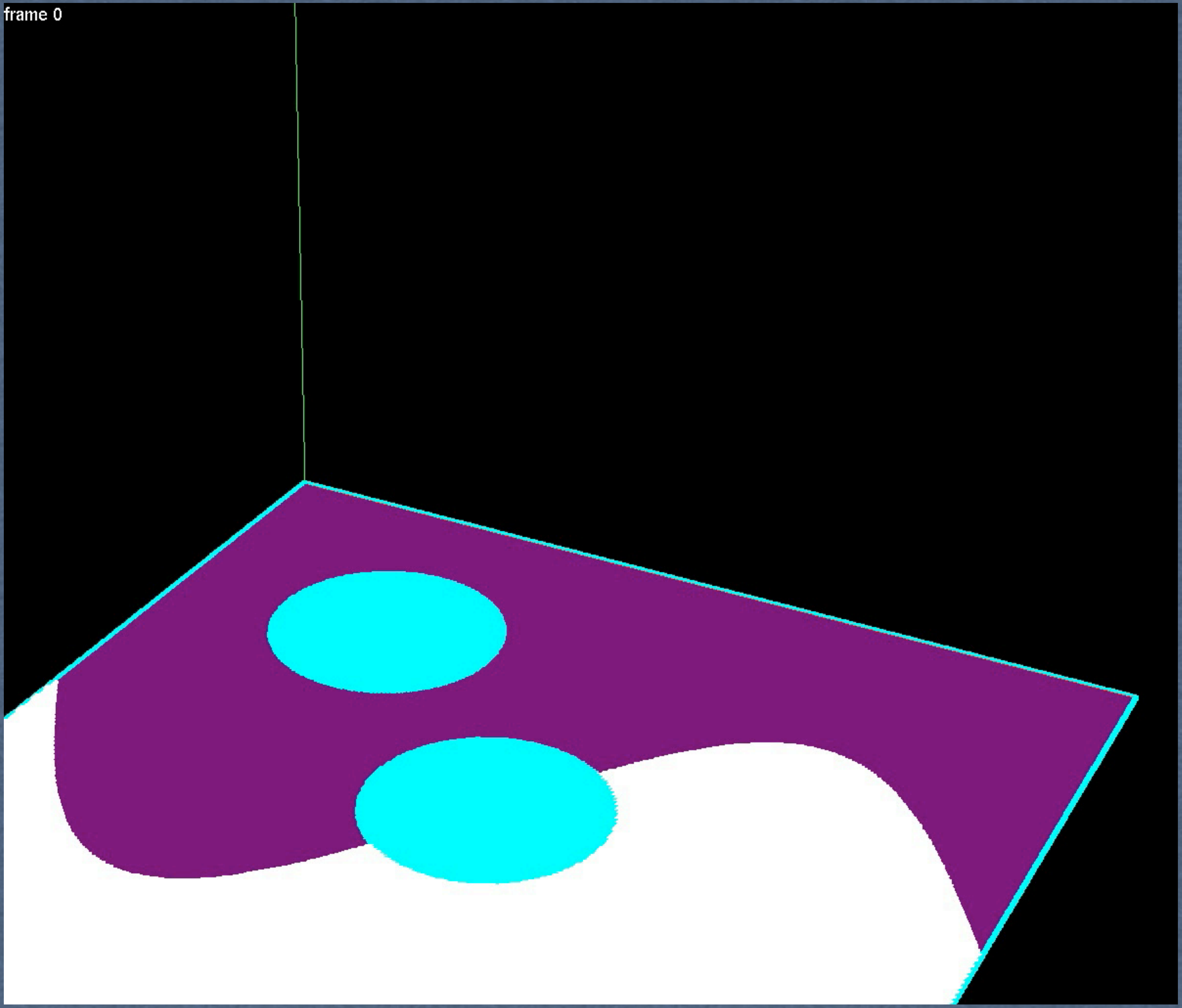
- Well suited for Poisson-type problems
- Should be able to provide resolution-independent convergence



Multigrid iterations towards convergence (1 V-Cycle/frame)



Multigrid iterations towards convergence (random initial guess)



Conjugate Gradients (with parallel multigrid preconditioner)

Multigrid

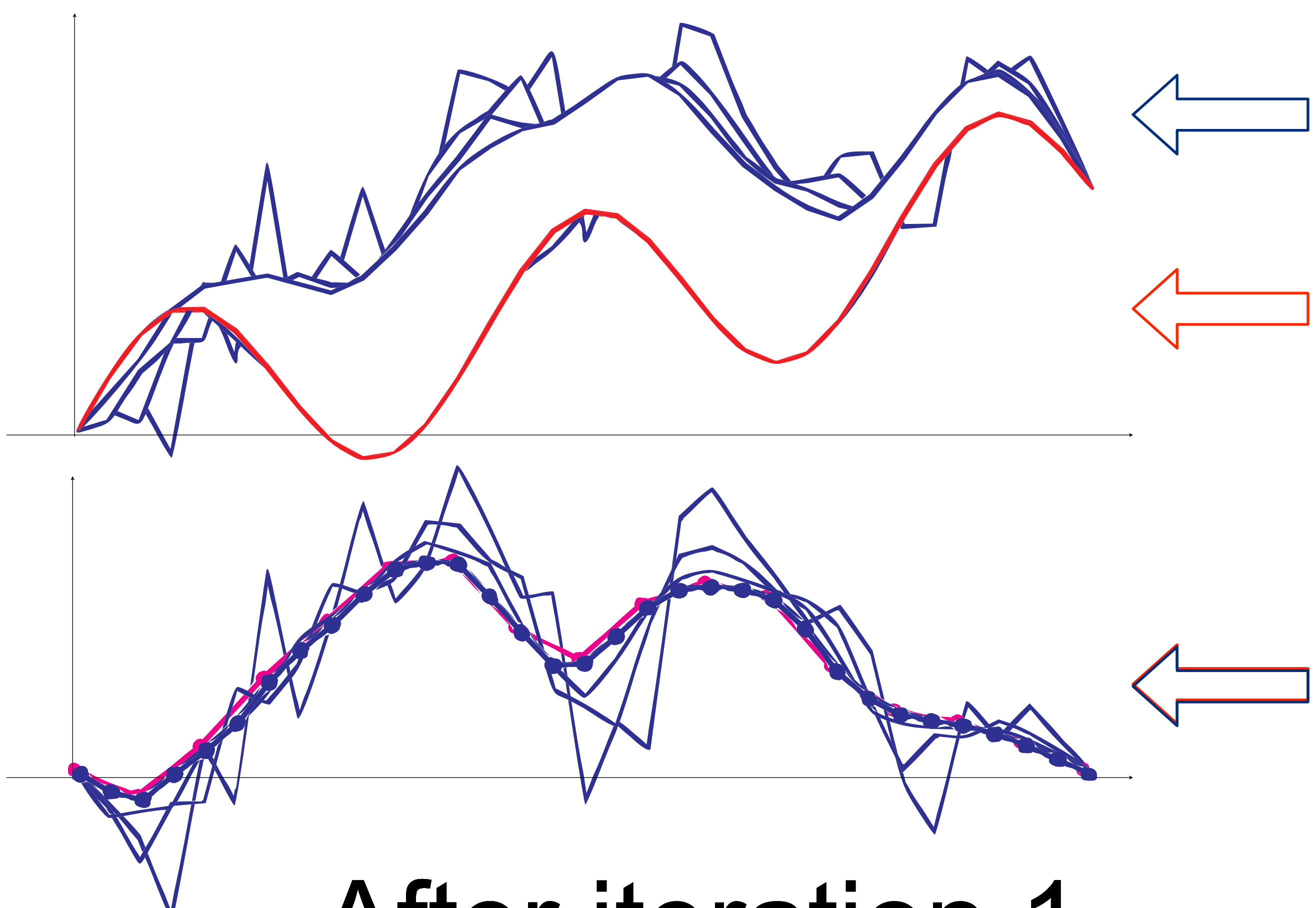
What about multigrid solvers?

- Well suited for Poisson-type problems
- Should be able to provide resolution-independent convergence

Multigrid: relaxation

$$\Delta u = f$$

$$u(0) = a, u(1) = b$$



After iteration 1

iteration 2

iteration 3

Relaxation:

a simple iterative solver

Initial guess

E.g., Gauss-Seidel

Exact solution

Fine level error

Coarse error

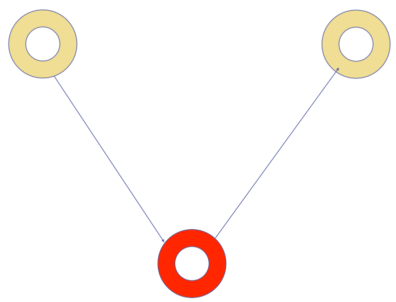
$e = u_{\text{current}} - u$

easier to solve

Error equation:

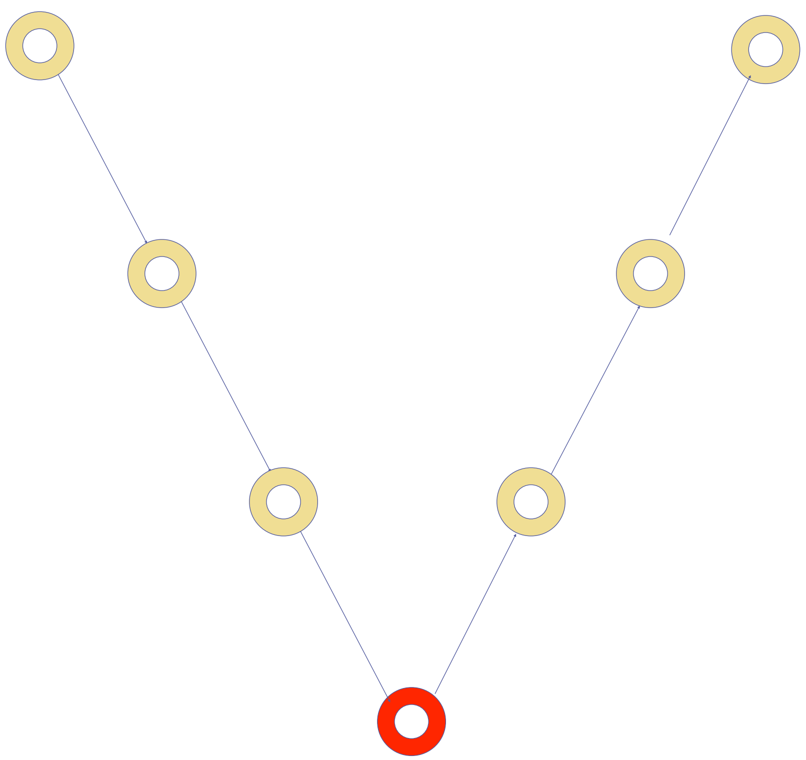
$$\Delta e = r \ (= \Delta u_{\text{current}} - f)$$

Multigrid cycle



- Relaxation
- Restriction
- Coarse grid solver
- Prolongation

Multigrid V-cycle



- Relaxation
- Restriction
- Coarse grid solver
- Prolongation

- Recursive coarsening
- Multigrid V-Cycle - O(N)
- Resolution independent numerical efficiency

Multigrid

What about multigrid solvers?

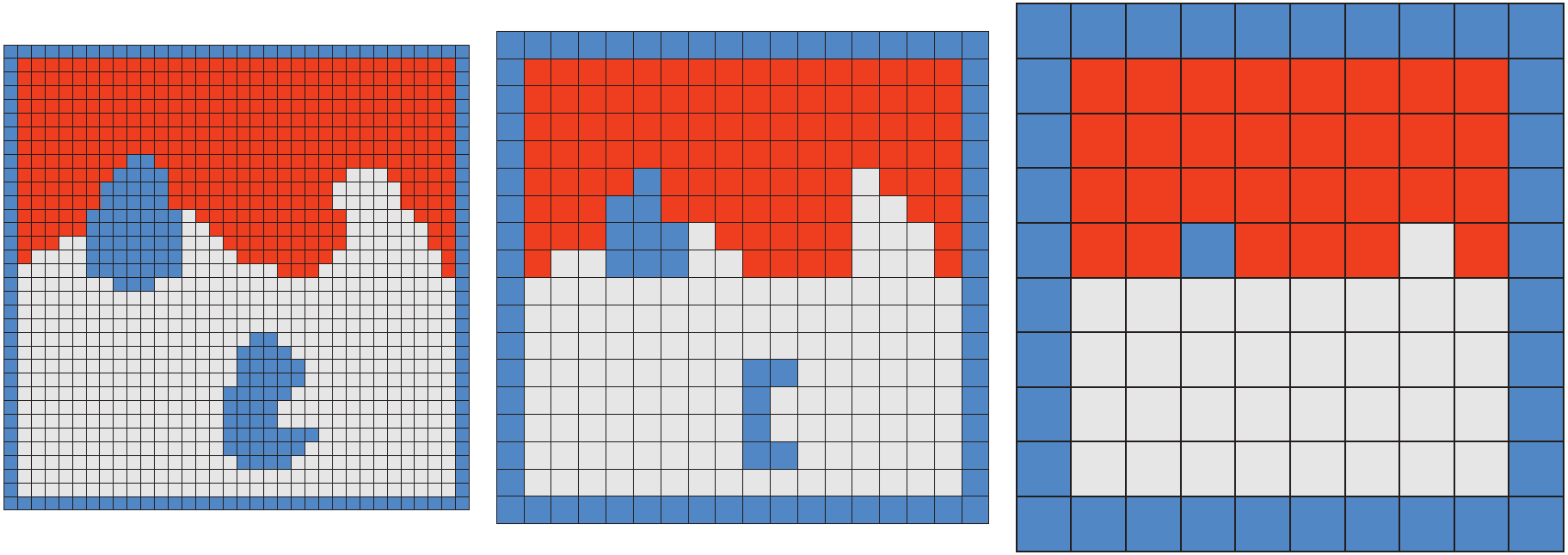
- Well suited for Poisson-type problems
- Should be able to provide resolution-independent convergence

For a multigrid solver, we need 3 algorithmic components

- A hierarchy of discretizations
- Transfer operators (restriction - prolongation)
- A "smoothing" routine

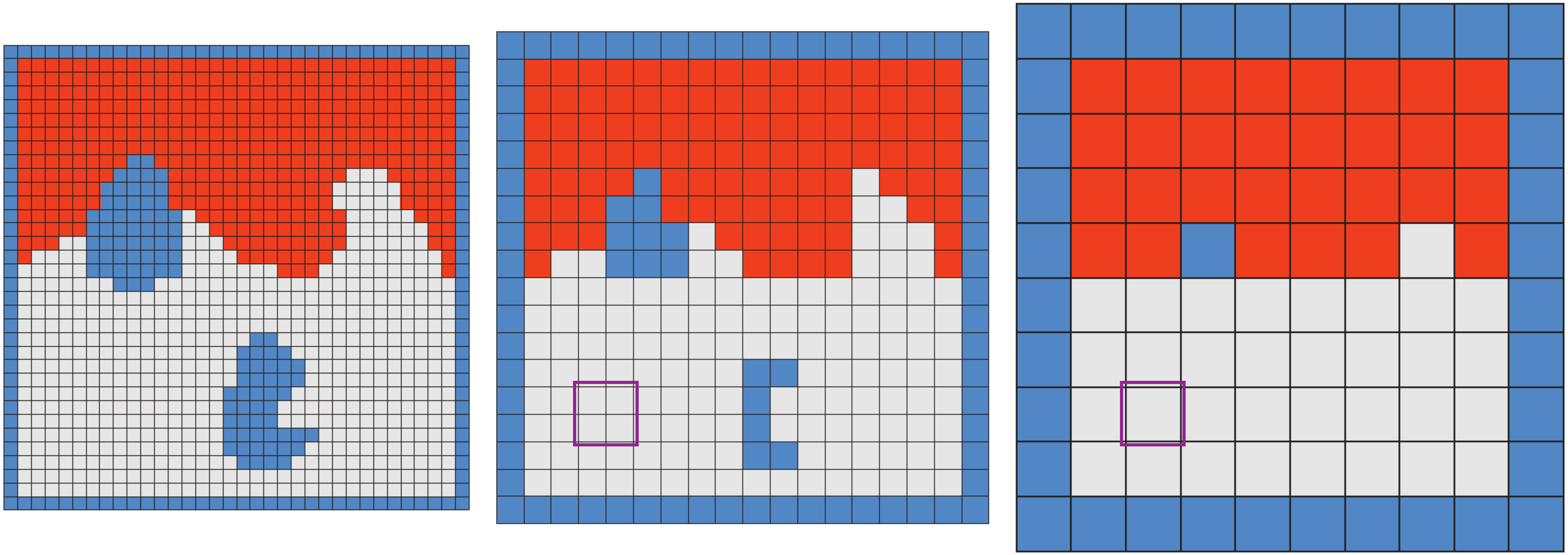
Multigrid

Creating a hierarchy of discretizations ...



Multigrid

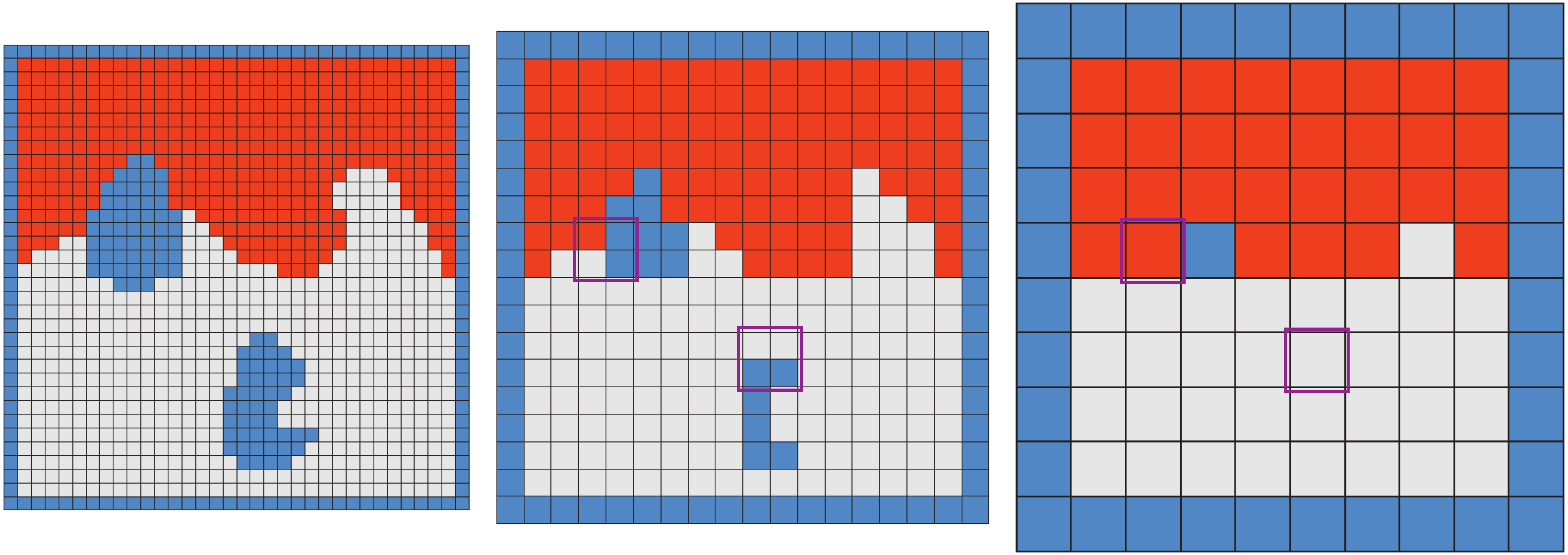
Creating a hierarchy of discretizations ...



Every group of 4 cells is coarsened into 1 cell of at the immediately coarser level

Multigrid

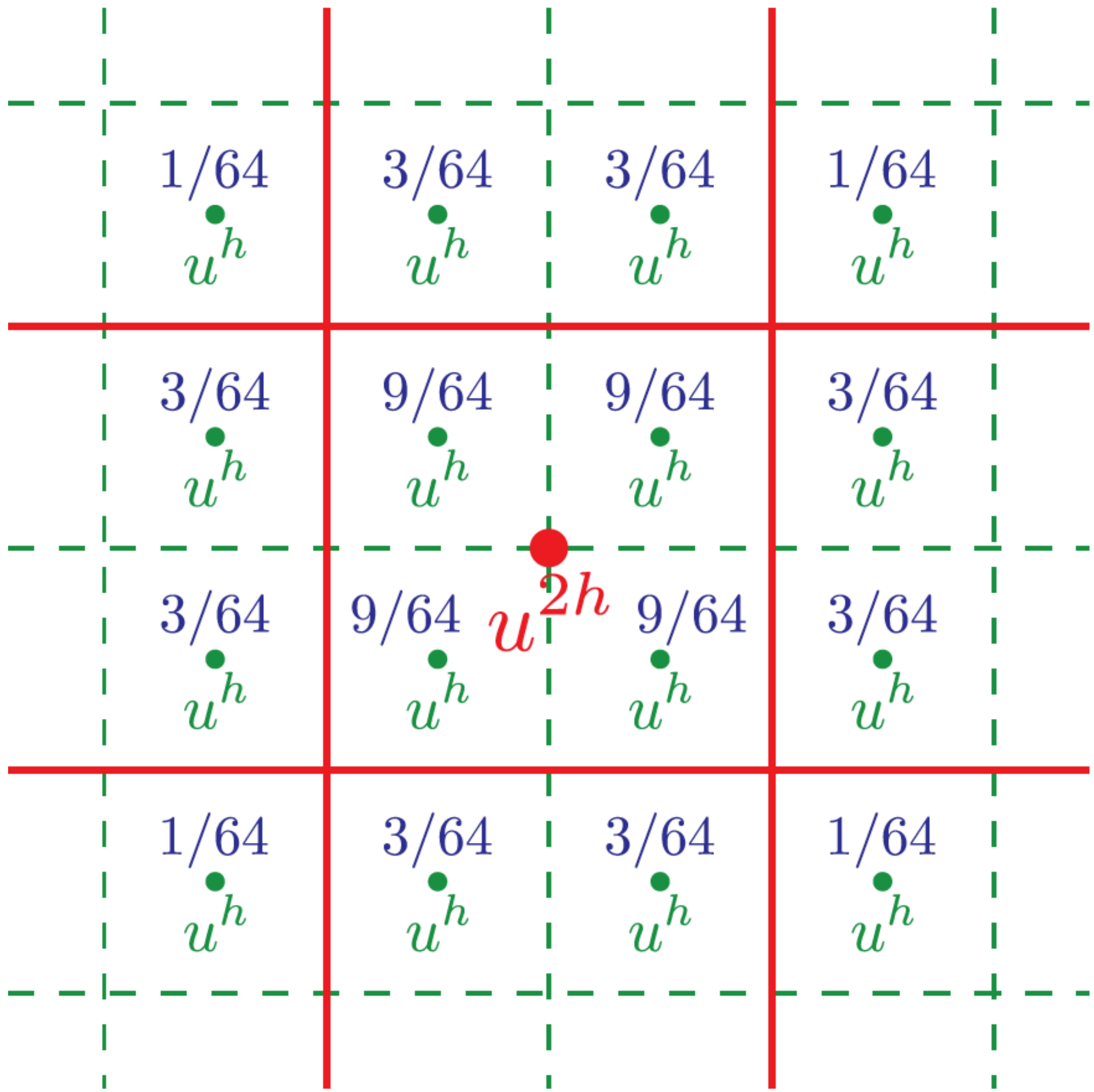
Creating a hierarchy of discretizations ...



When fine grid cells have more than one type, the coarse cell becomes (by priority) Dirichlet -- Interior (if no Dirichlet) -- Neumann (if no Interior or Dirichlet)

Multigrid

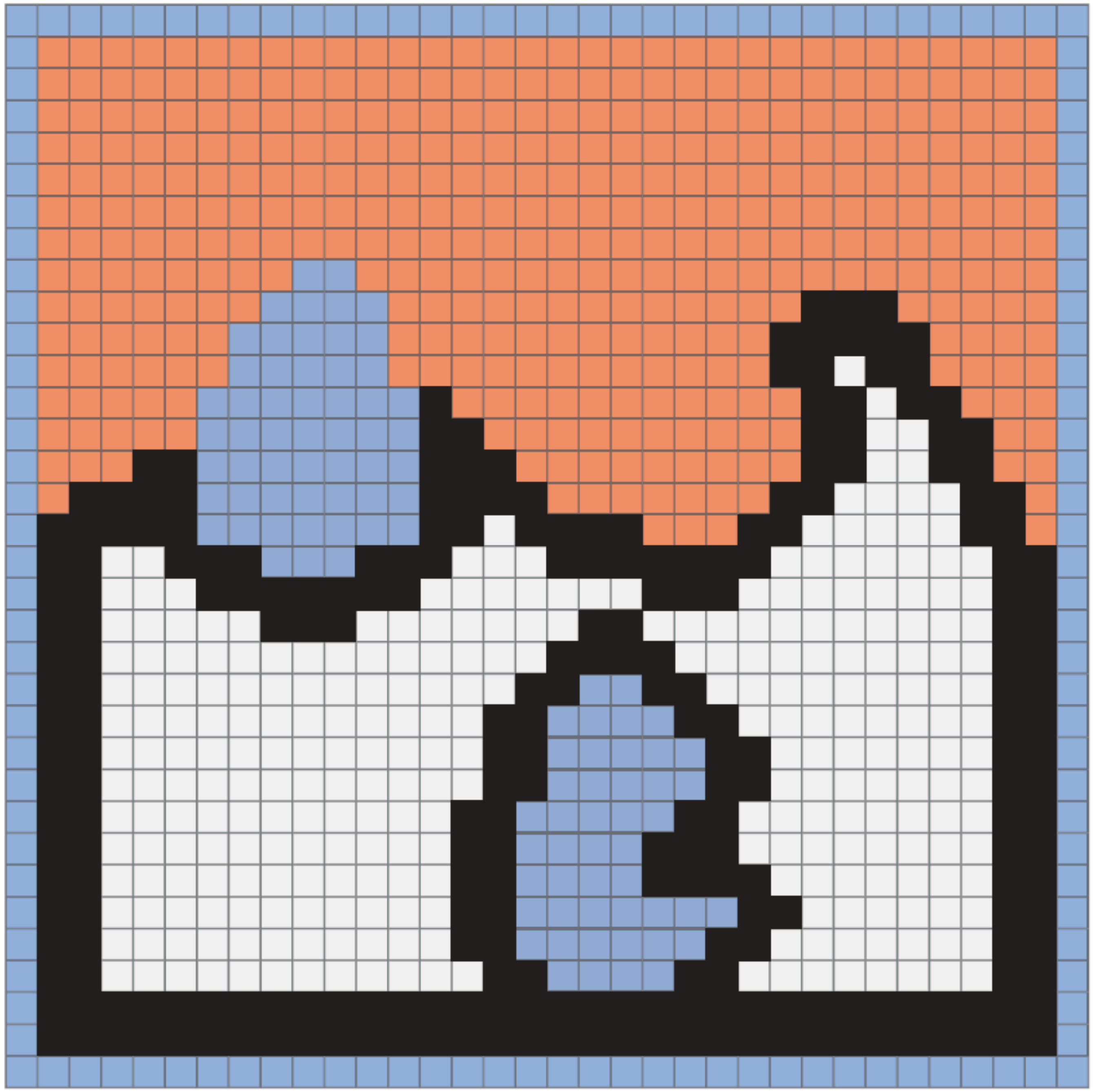
Inter-grid transfer operators (Prolongation - Restriction)

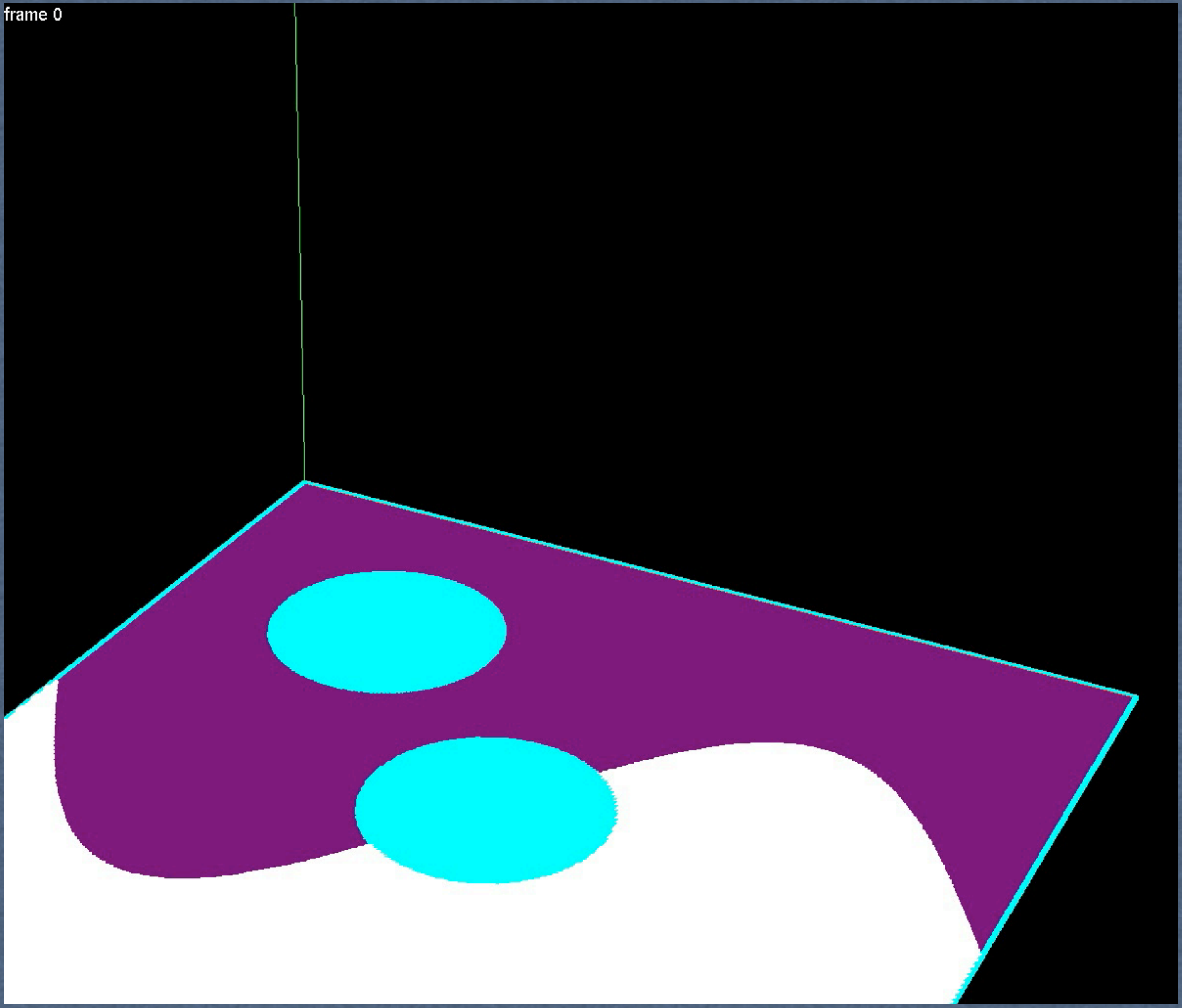


Multigrid

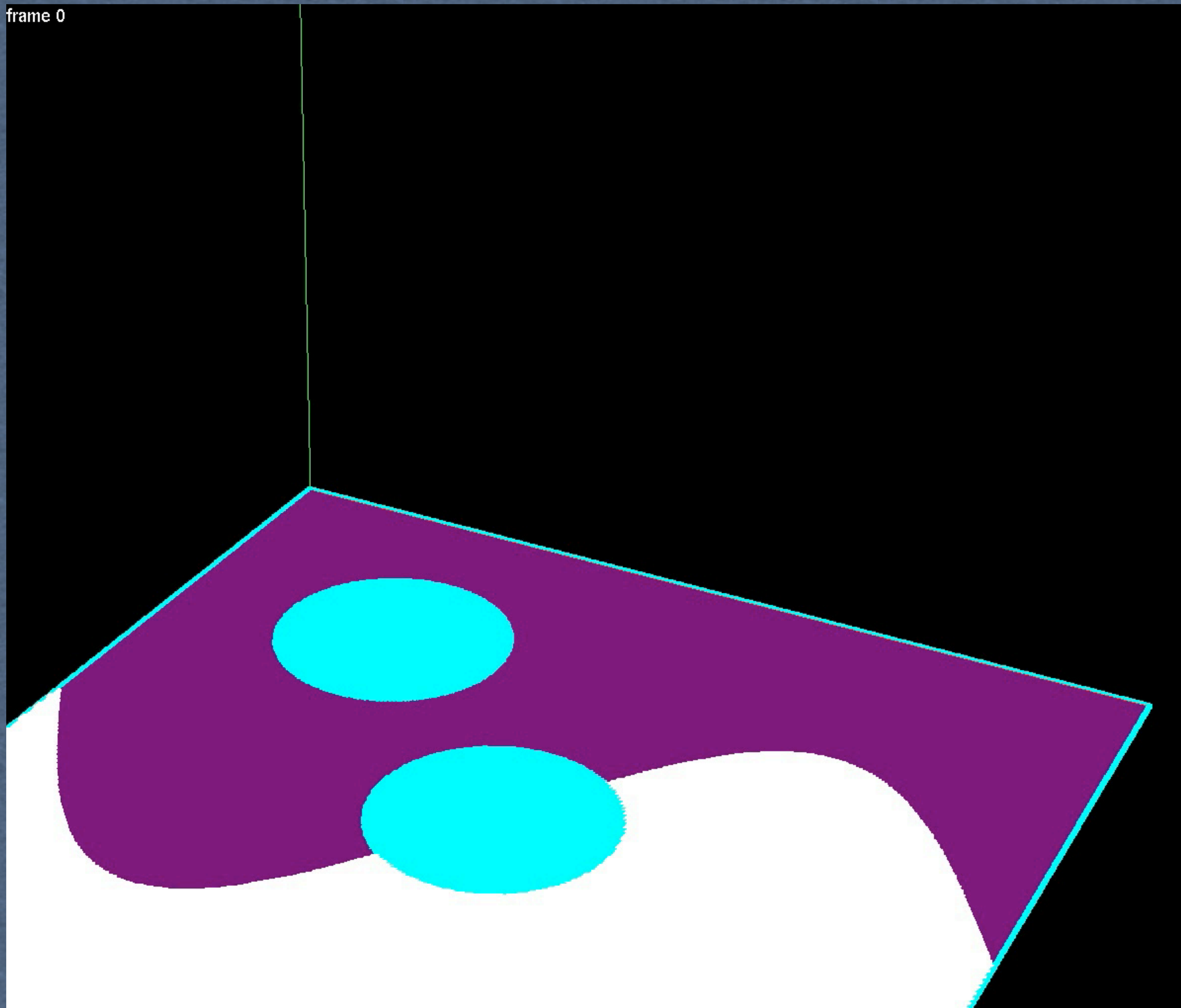
Smoothing procedure :

- *Perform N sweeps of relaxation (e.g. Gauss-Seidel method) on a band around the boundary*
- *Perform 1 sweep in the interior*
- *Perform N more sweeps on the boundary band*

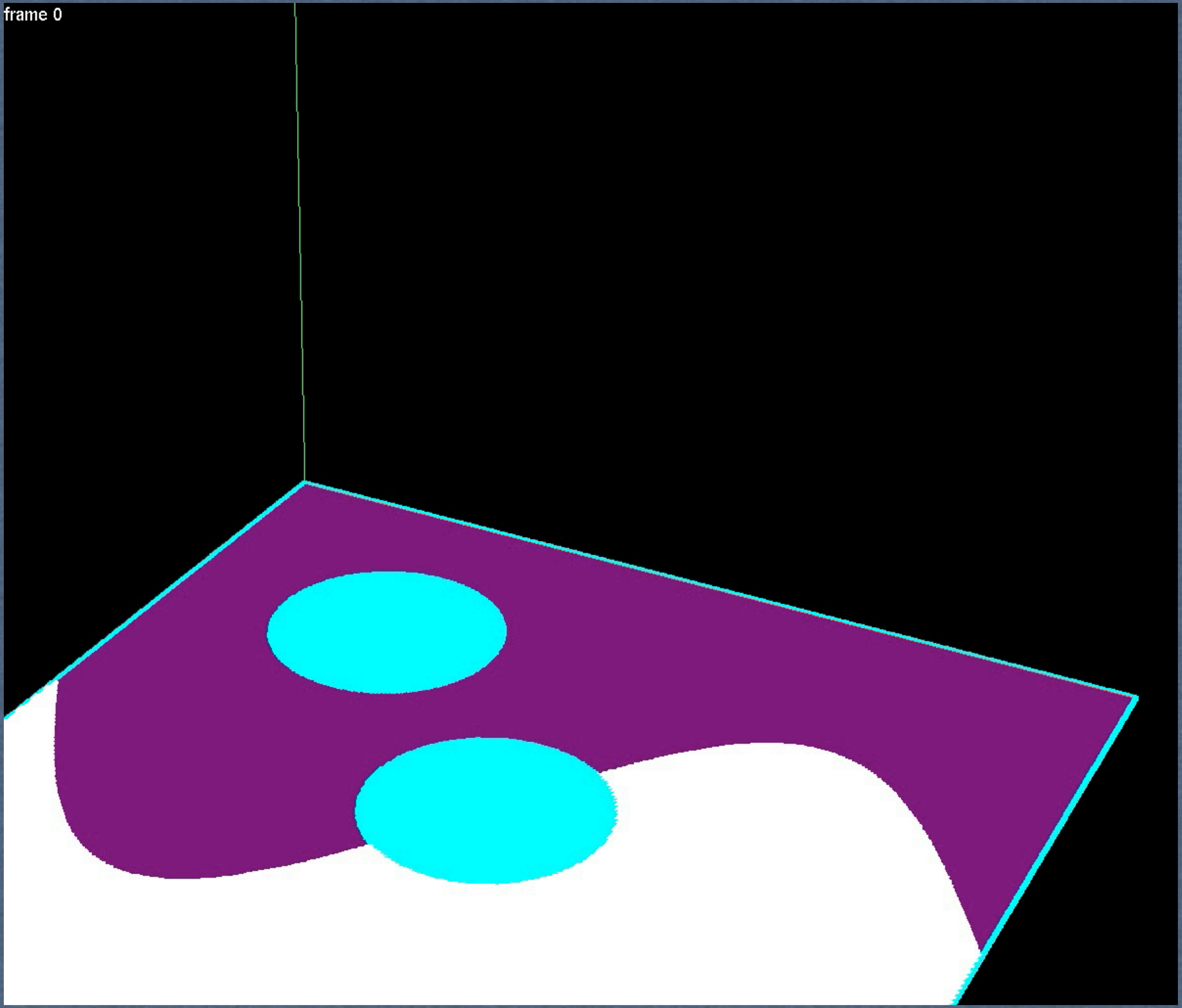




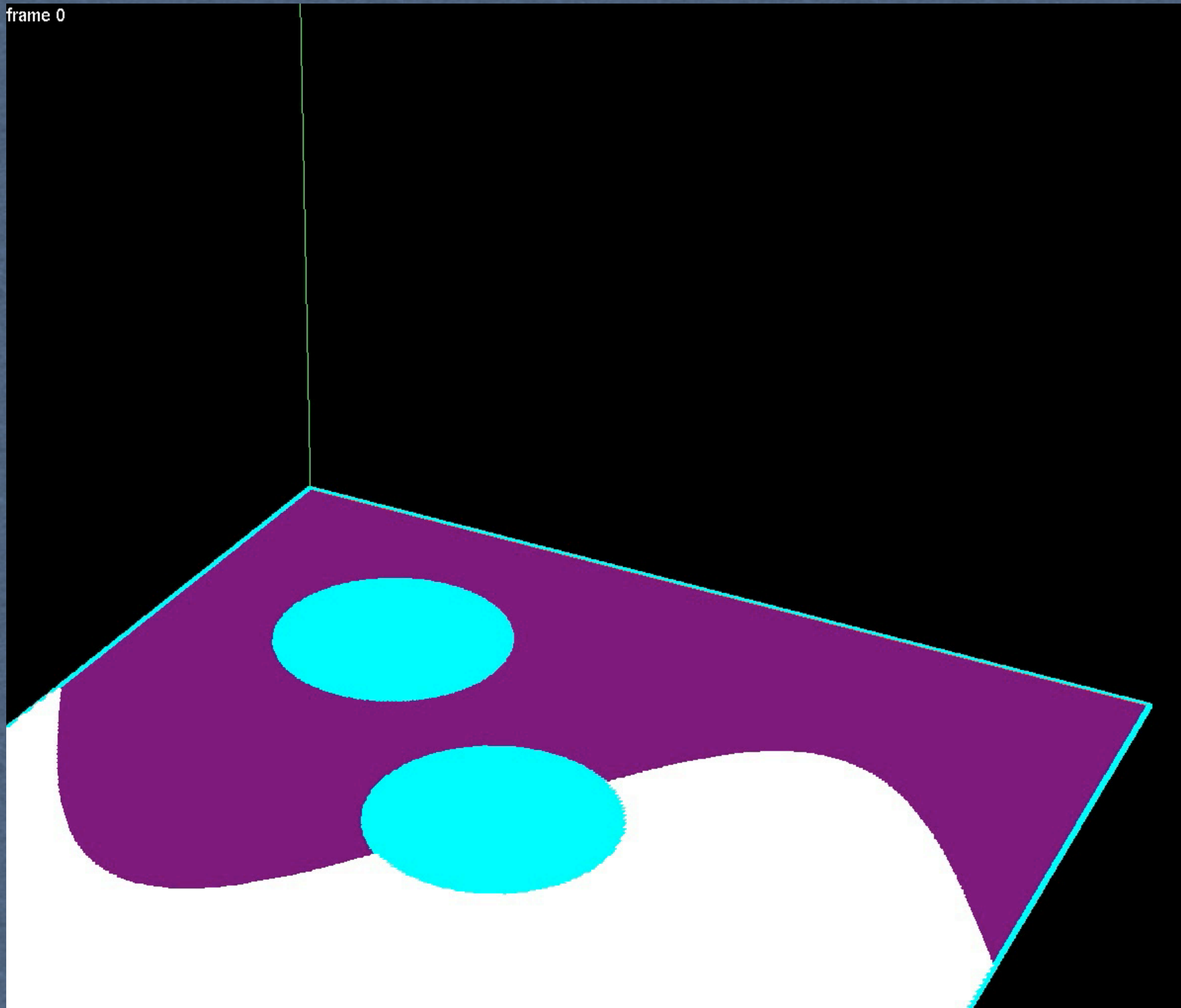
Stable Multigrid V-cycle (30 boundary smoothing sweeps)



MG-PCG (same boundary smoothing effort as stable V-cycle)



MG-PCG (1/3 of the smoothing effort needed for stable V-cycle)



MG-PCG (1/30 of the smoothing effort needed for stable V-cycle)

Multigrid

No free lunch ...

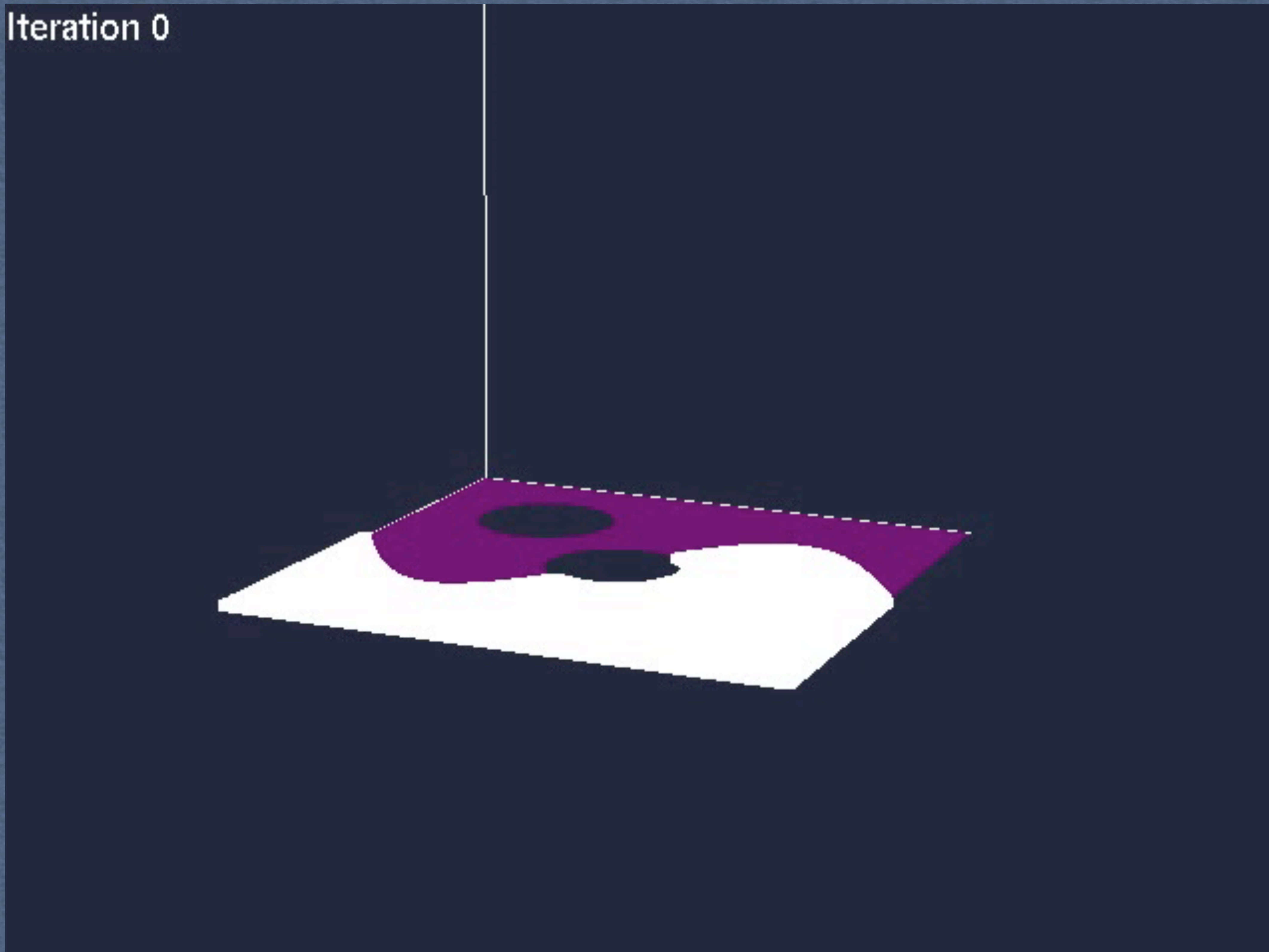
- Convergence & Stability may necessitate an impractically intensive boundary smoothing effort

N = 30 for stability !

Smoothing procedure :

- *Perform N sweeps of relaxation (e.g. Gauss-Seidel method) on a band around the boundary*
- *Perform 1 sweep in the interior*
- *Perform N more sweeps on the boundary band*

N



V-Cycle iteration, 10 boundary iterations per cycle (1/3 of the effort needed for convergent method)

Multigrid

No free lunch ...

- Convergence & Stability may necessitate an impractically intensive boundary smoothing effort
- For moderate resolutions (e.g. 128^3) one V-cycle iteration may cost as much as 20-30 iterations of (unpreconditioned) CG

Multigrid

No free lunch ...

- Convergence & Stability may necessitate an impractically intensive boundary smoothing effort
- For moderate resolutions (e.g. 128^3) one V-cycle iteration may cost as much as 20-30 iterations of (unpreconditioned) CG
- Boundary cost goes away *asymptotically*, but only at impractically high resolutions

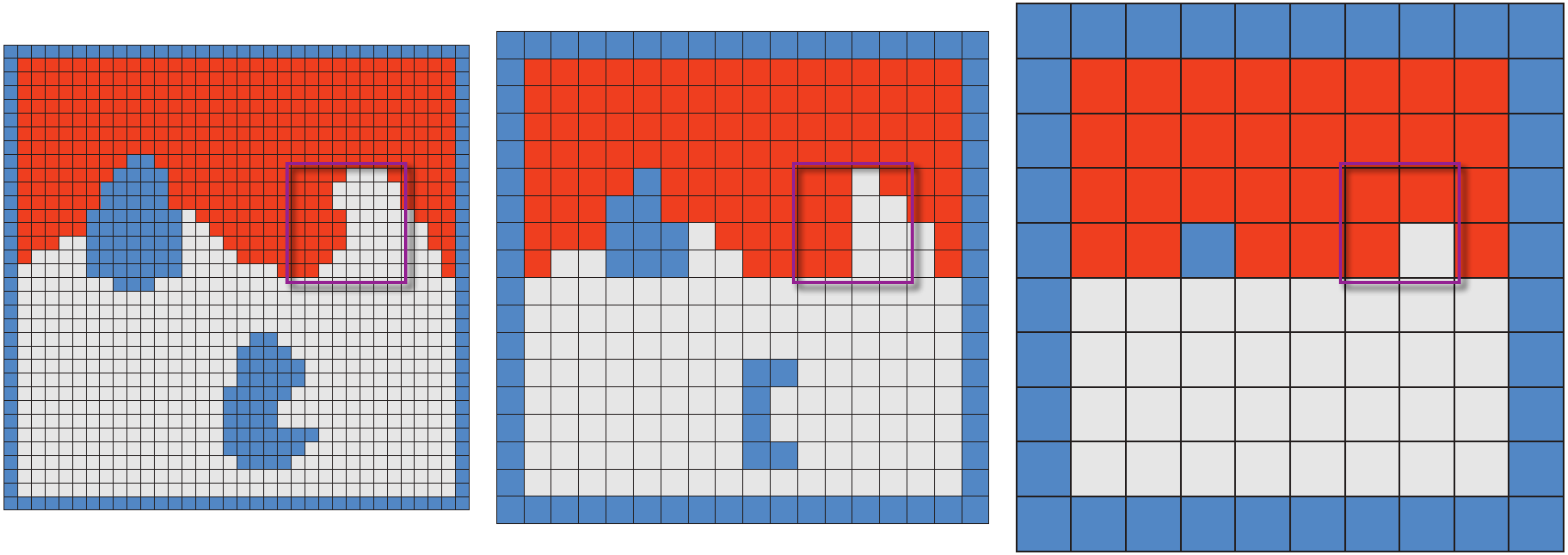
Multigrid

No free lunch ...

- Convergence & Stability may necessitate an impractically intensive boundary smoothing effort
- For moderate resolutions (e.g. 128^3) one V-cycle iteration may cost as much as 20-30 iterations of (unpreconditioned) CG
- Boundary cost goes away *asymptotically*, but only at impractically high resolutions
- Even worse, when considering parallelism

Multigrid

Problem #1 : Geometric discrepancies lead to instability

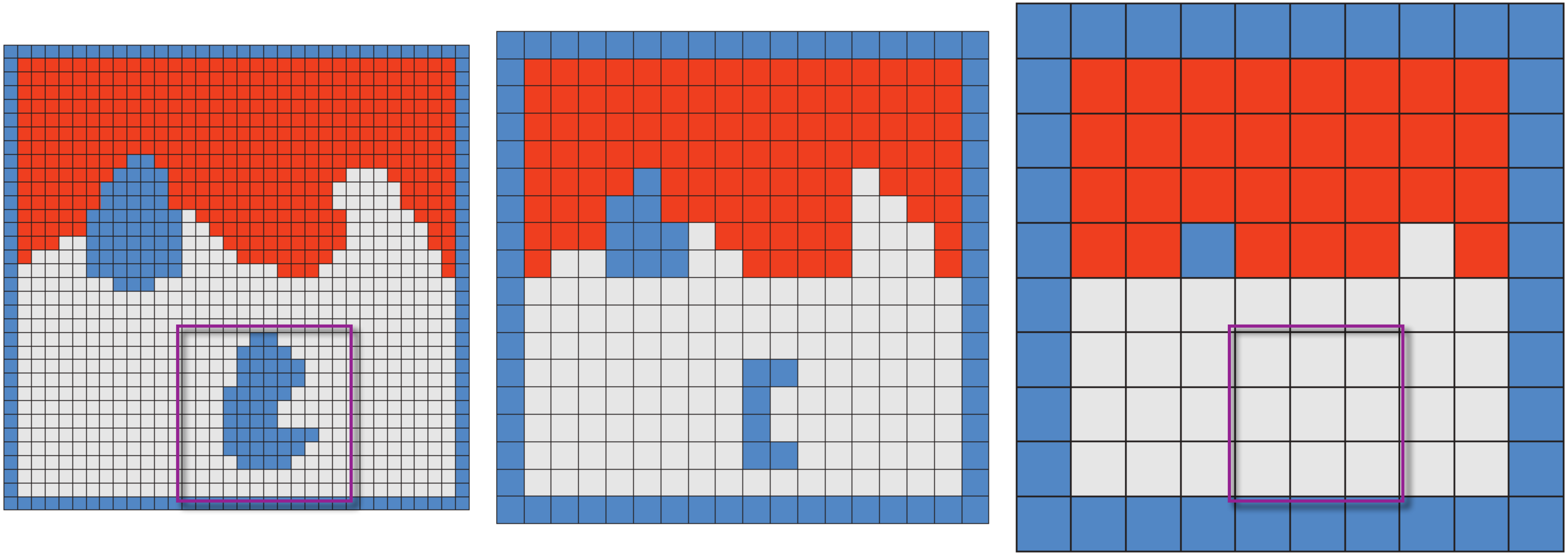


Remedies : Intensive boundary smoothing, algebraic coarsening, specialized transfer operators …

… or using MG as a preconditioner for a Krylov method

Multigrid

Problem #2 : Topological discrepancies lead to stagnation



Remedies : Algebraic coarsening, recombined iterants,
using a fully Algebraic Multigrid method ...
... or using MG as a preconditioner for a Krylov method

Multigrid preconditioning

Basic concepts of Preconditioned Conjugate Gradients :

- Basic problem : $\mathbf{A}\mathbf{x} = \mathbf{b}$

Multigrid preconditioning

Basic concepts of Preconditioned Conjugate Gradients :

- Basic problem : $\mathbf{Ax} = \mathbf{b}$
- Consider a matrix $\mathbf{M}$ such that $\mathbf{MA} \approx \mathbf{I}$

Multigrid preconditioning

Basic concepts of Preconditioned Conjugate Gradients :

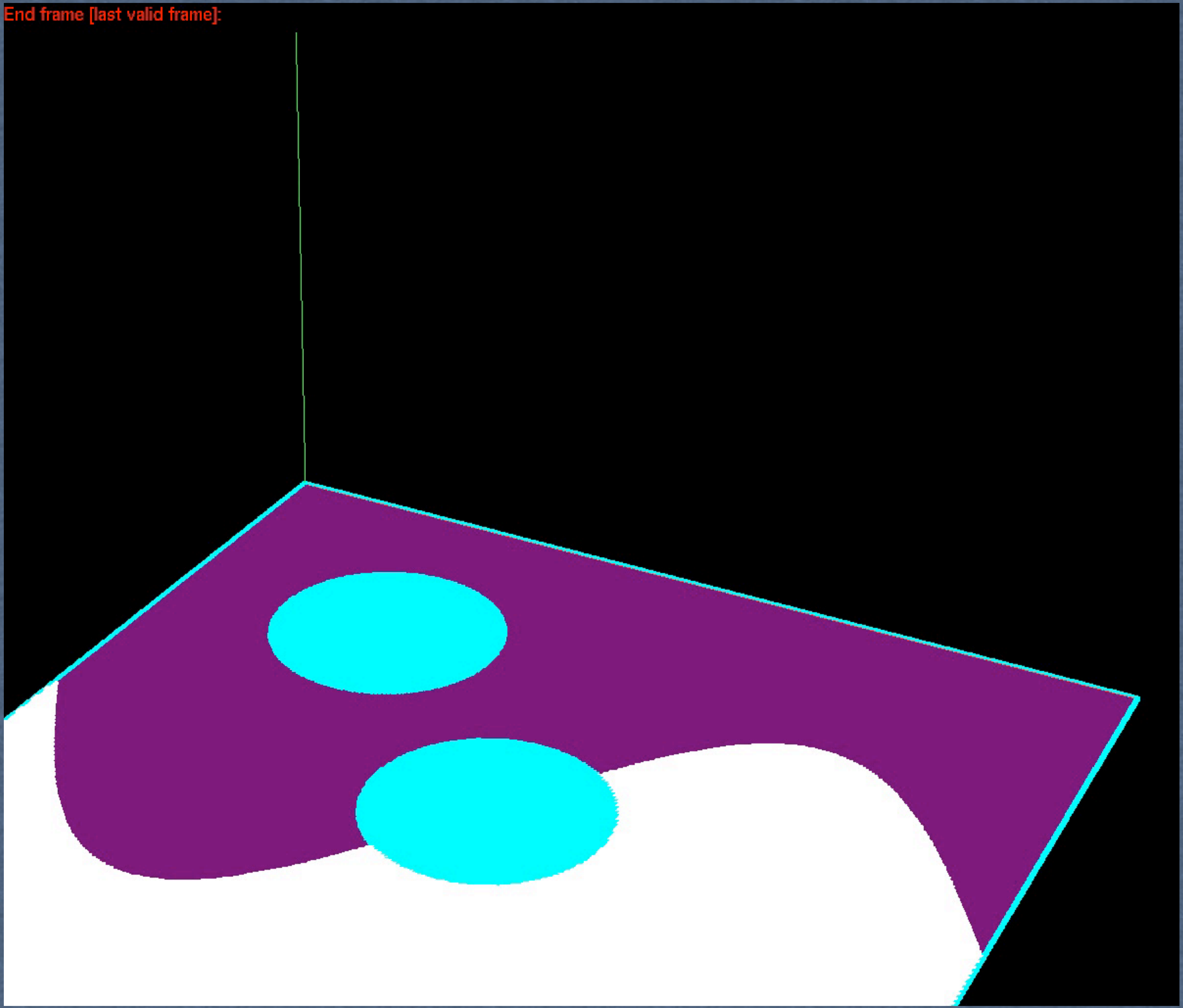
- Basic problem : $\mathbf{Ax} = \mathbf{b}$
- Consider a matrix $\mathbf{M}$ such that $\mathbf{MA} \approx \mathbf{I}$
- This matrix needs to be symmetric and positive definite.
  In this case we can write : $\mathbf{M} = \mathbf{U}^2$ where $\mathbf{U}$ is symmetric

Multigrid preconditioning

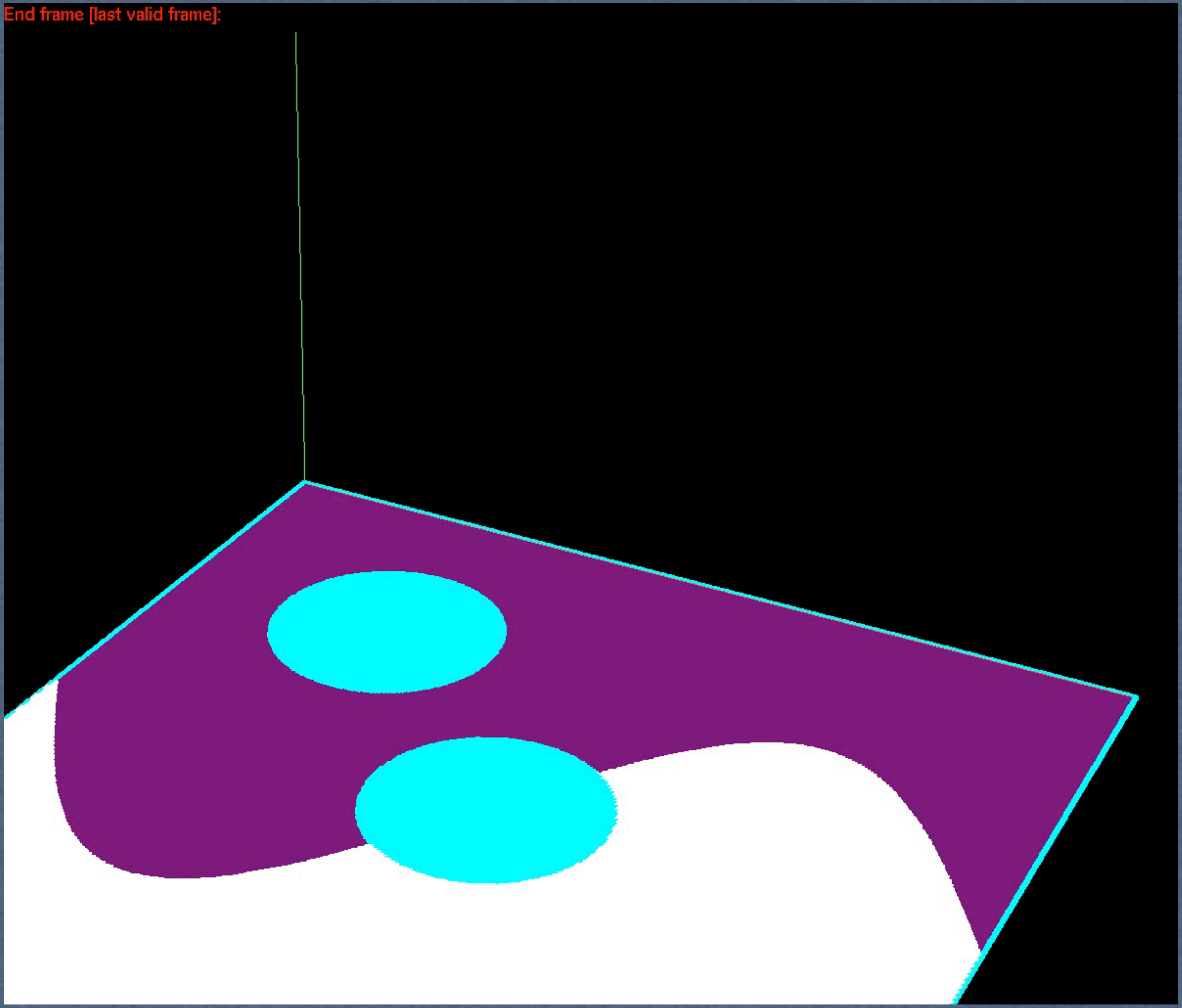
Basic concepts of Preconditioned Conjugate Gradients :

- Basic problem : $\mathbf{Ax} = \mathbf{b}$
- Consider a matrix $\mathbf{M}$ such that $\mathbf{MA} \approx \mathbf{I}$
- This matrix needs to be symmetric and positive definite.
  In this case we can write : $\mathbf{M} = \mathbf{U}^2$ where $\mathbf{U}$ is symmetric
- The PCG method is algebraically equivalent to applying the Conjugate Gradients method on the modified system :

$$(\mathbf{UAU})(\mathbf{U}^{-1}\mathbf{x}) = \mathbf{Ub}$$



Conjugate Gradients (w/o preconditioning)



Conjugate Gradients (with a stock preconditioner)

Preconditioned Conjugate Gradients

procedure MGPCG($\mathbf{r}, \mathbf{x}$)
  $\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
  if $(\nu < \nu_{\max})$ **then return**
  $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
  for $k = 0$ **to** k_{max} **do**
    $\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
    $\alpha \leftarrow \rho/\sigma$
    $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
    if $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
      $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
      return
    end if
    $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
    $\beta \leftarrow \rho^{\text{new}}/\rho$
    $\rho \leftarrow \rho^{\text{new}}$
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
  end for
end procedure

Multigrid preconditioning

Basic concepts of Preconditioned Conjugate Gradients :

- Basic problem : $\mathbf{A}\mathbf{x} = \mathbf{b}$
- Consider a matrix $\mathbf{M}$ such that $\mathbf{M}\mathbf{A} \approx \mathbf{I}$
- This matrix needs to be symmetric and positive definite.
  In this case we can write : $\mathbf{M} = \mathbf{U}^2$ where $\mathbf{U}$ is symmetric
- The PCG method is algebraically equivalent to applying the Conjugate Gradients method on the modified system :

$$(\mathbf{U}\mathbf{A}\mathbf{U})(\mathbf{U}^{-1}\mathbf{x}) = \mathbf{U}\mathbf{b}$$

- The final algorithm requires only a routine for computing $\mathbf{M}\mathbf{v}$ (for any input vector) to be specified

Multigrid preconditioning

Using a multigrid V-cycle to define a preconditioner :

Multigrid preconditioning

Using a multigrid V-cycle to define a preconditioner :

- Define a hierarchy of discrete Poisson problems ***with zero Dirichlet and Neumann Boundary conditions at all levels.***

Multigrid preconditioning

Using a multigrid V-cycle to define a preconditioner :

- Define a hierarchy of discrete Poisson problems ***with zero Dirichlet and Neumann Boundary conditions at all levels***.
- Use the input vector ***v*** as the right hand side at the top level. ***Use a zero initial guess at every level of the V-cycle***.

Multigrid preconditioning

Using a multigrid V-cycle to define a preconditioner :

- Define a hierarchy of discrete Poisson problems ***with zero Dirichlet and Neumann Boundary conditions at all levels***.

- Use the input vector ***v*** as the right hand side at the top level. ***Use a zero initial guess at every level of the V-cycle***.

- Perform 1 V-cycle. The result is the desired product ***Mv.***

Multigrid preconditioning

Using a multigrid V-cycle to define a preconditioner :

- Define a hierarchy of discrete Poisson problems ***with zero Dirichlet and Neumann Boundary conditions at all levels***.
- Use the input vector ***v*** as the right hand side at the top level. ***Use a zero initial guess at every level of the V-cycle***.
- Perform 1 V-cycle. The result is the desired product ***Mv.***

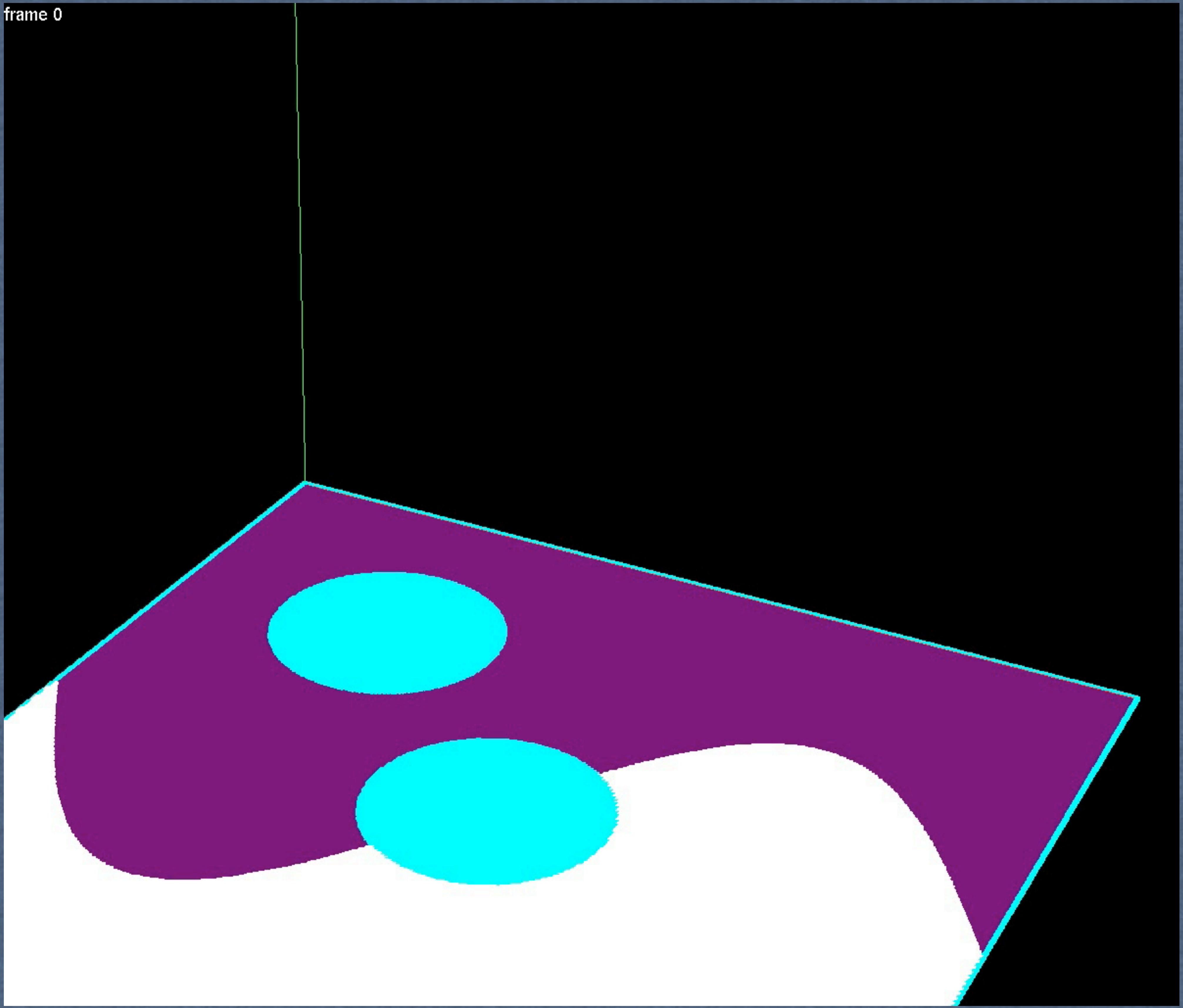
A valid CG preconditioner (symmetric, definite) is obtained if :

- Restriction - prolongation are defined as adjoint operators.
- Jacobi (or damped Jacobi) is used instead of Gauss-Seidel to relax the interior of the domain.
- Boundary band is traversed by the smoother in opposite orders during the downstroke and upstroke of the V-cycle.

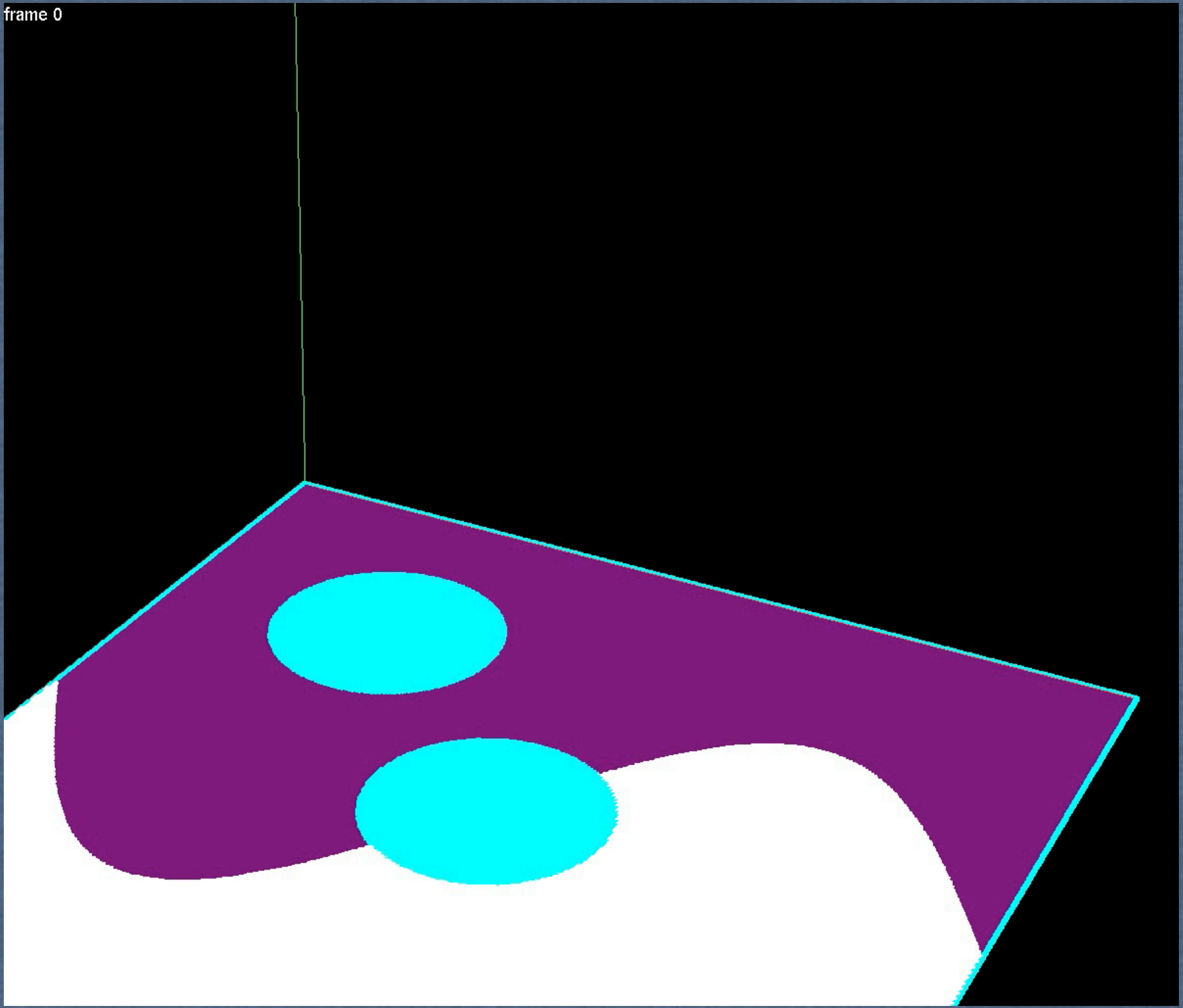
Multigrid preconditioning

MG-preconditioned CG benefits :

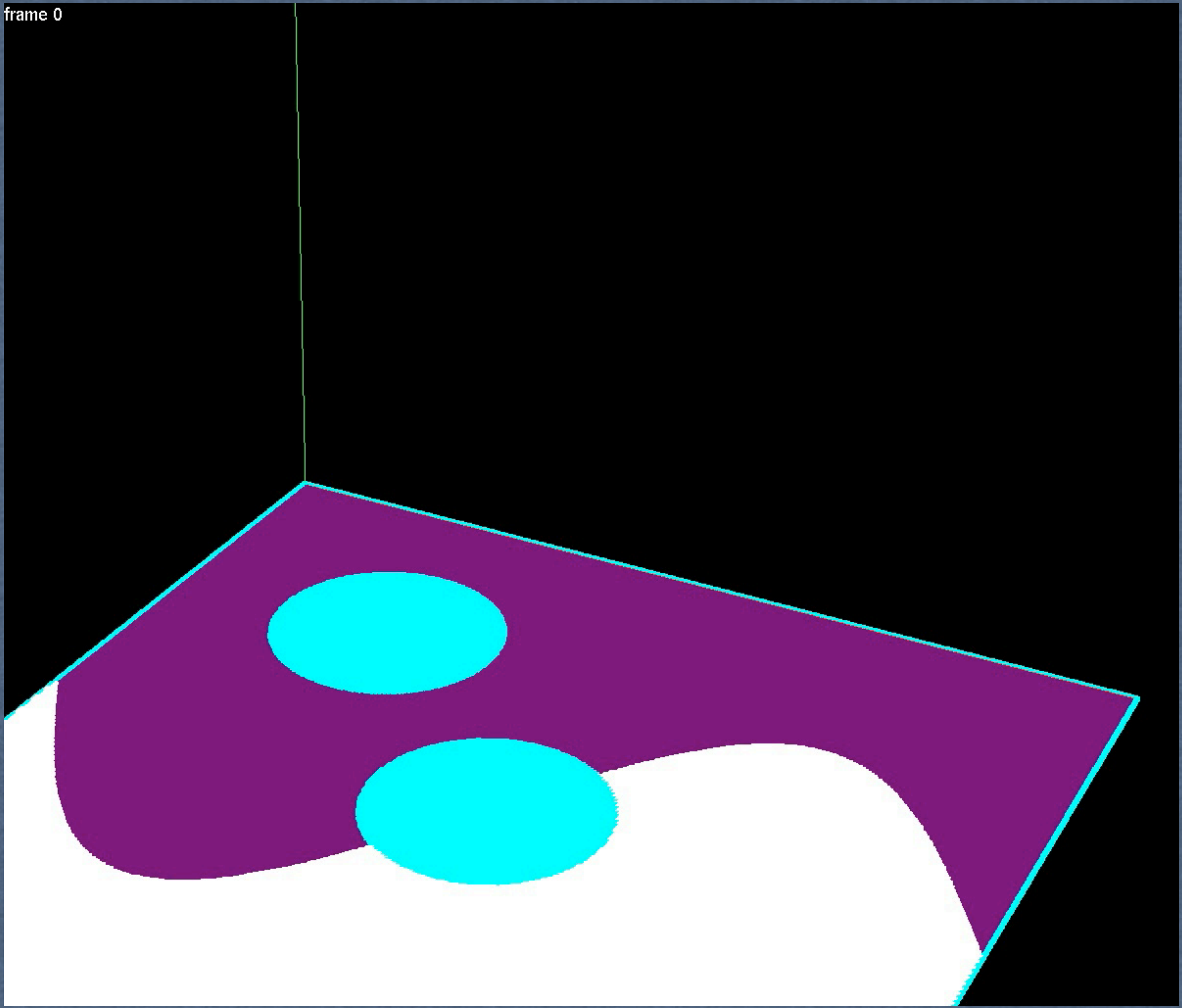
- Remains stable, even if an ***unstable*** V-cycle is used for preconditioning



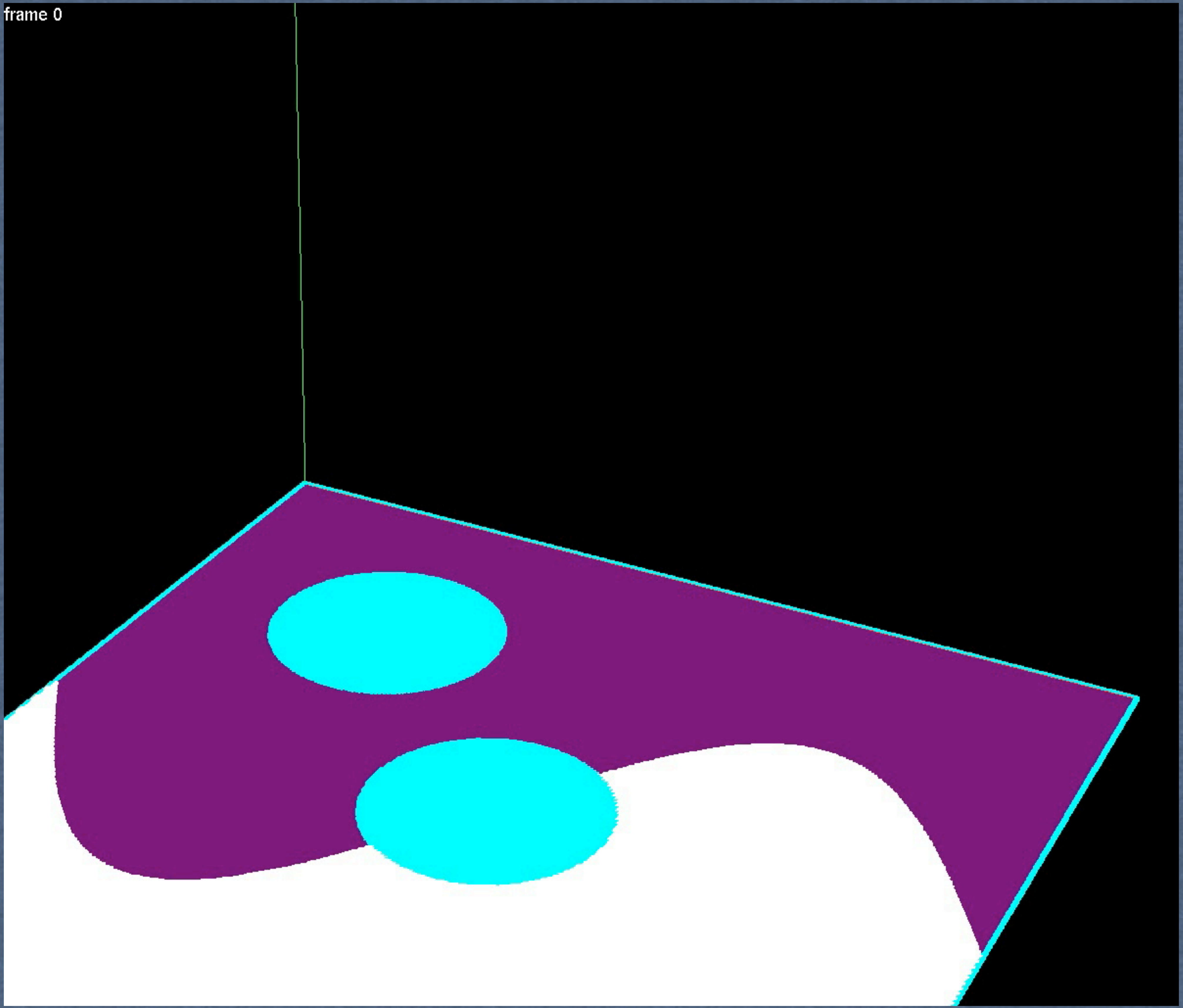
Stable Multigrid V-cycle (30 boundary smoothing sweeps)



MG-PCG (same boundary smoothing effort as stable V-cycle)



MG-PCG (1/3 of the smoothing effort needed for stable V-cycle)



MG-PCG (1/30 of the smoothing effort needed for stable V-cycle)

Multigrid preconditioning

MG-preconditioned CG benefits :

- Remains stable, even if an ***unstable*** V-cycle is used for preconditioning
- Unstable V-cycles simply require a few extra PCG iterations

Multigrid preconditioning

MG-preconditioned CG benefits :

- Remains stable, even if an ***unstable*** V-cycle is used for preconditioning
- Unstable V-cycles simply require a few extra PCG iterations
- Intensity of the boundary treatment can be tuned to ***moderate-difficulty scenarios***, and remain stable even in highly complicated cases

Multigrid preconditioning

MG-preconditioned CG benefits :

- Remains stable, even if an ***unstable*** V-cycle is used for preconditioning
- Unstable V-cycles simply require a few extra PCG iterations
- Intensity of the boundary treatment can be tuned to ***moderate-difficulty scenarios***, and remain stable even in highly complicated cases
- With a well-designed (albeit unstable) V-cycle, PCG converges as quickly as the best-case scenario multigrid cycle, in practice.

Results and performance