Welcome!

Lecture 8, 10 - “Variance Reduction”
Today's Agenda:

- Introduction
- Stratification
- Next Event Estimation
- Importance Sampling
- Resampled Importance
Introduction

Previously in Advanced Graphics
Introduction
Introduction

Today in Advanced Graphics:

- Stratification
- Next Event Estimation
- Importance Sampling
- Multiple Importance Sampling
- Resampled Importance Sampling*

Aim:

- to get a better image with the same number of samples
- to increase the efficiency of a path tracer
- to reduce variance in the estimate

Requirement:

- produce the correct image

*: If time permits
Today's Agenda:

- Introduction
- Stratification
- Next Event Estimation
- Importance Sampling
- Resampled Importance
Stratification

Uniform Random Sampling

To sample a light source, we draw two random values in the range 0..1.

The resulting 2D positions are not uniformly distributed over the area.

We can improve uniformity using *stratification*: one sample is placed in each stratum.

```plaintext
// ...

int N = 100; // Number of samples

for (int i = 0; i < N; i++) {
    // ...}
```
Uniform Random Sampling

To sample a light source, we draw two random values in the range 0..1.

The resulting 2D positions are not uniformly distributed over the area.

We can improve uniformity using **stratification**: one sample is placed in each stratum.

For 4x4 strata:

\[
\text{stratum}_x = (\text{idx} \mod 4) \times 0.25 \quad // \text{idx} = 0..15 \\
\text{stratum}_y = (\text{idx} / 4) \times 0.25
\]

\[
\begin{align*}
r0 &= \text{Rand()} \times 0.25 \\
r1 &= \text{Rand()} \times 0.25 \\
P &= \text{vec2}(\text{stratum}_x + r0, \text{stratum}_y + r1)
\end{align*}
\]
Stratification

Advanced Graphics – Variance Reduction
Stratification can be applied to any Monte Carlo process:

- Anti-aliasing (sampling the pixel)
- Depth of field (sampling the lens)
- Motion blur (sampling time)
- Soft shadows (sampling area lights)
- Diffuse reflections (sampling the hemisphere)

However, there are problems:

- We need to take one sample per stratum
- Stratum count: higher is better, but with diminishing returns
- Combining stratification for e.g. depth of field and soft shadows leads to correlation of the samples, unless we stratify the 4D space - which leads to a very large number of strata: the **curse of dimensionality**.
Stratification

Troubleshooting Path Tracing Experiments

When experimenting with stratification and other variance reduction methods you will frequently produce incorrect images.

Tip:

Keep a simple reference path tracer without any tricks. Compare your output to this reference solution frequently.
Today's Agenda:

- Introduction
- Stratification
- Next Event Estimation
- Importance Sampling
- Resampled Importance
Next Event Estimation

Recall the rendering equation: 
\[ L_o(x, \omega_o) = L_E(x, \omega_o) + \int_{\Omega} f_r(x, \omega_o, \omega_i) L_i(x, \omega_i) \cos \theta_i \, d\omega_i \]

Also recall that we had two ways to sample direct illumination: 
- Integrating over the hemisphere
- Integrating over the lights

Can we apply this to the full rendering equation, instead of just direct illumination?

next_event_estimation.png

```
// integrating over the hemisphere
\[
L_o(p, \omega_o) = \int_{\Omega} f_r(p, \omega_o, \omega_i) L_d(p, \omega_i) \cos \theta_i \, d\omega_i
\]

// integrating over the lights
\[
L_o(p, \omega_i) = \sum_{j=1}^{lights} \int_{A} f_r(p, \omega_o, \omega_i) L_l^d(p, \omega_i) \cos \theta_i \, d\omega_i
\]

Can we apply this to the full rendering equation, instead of just direct illumination?
Incoming direct light

\[
L_d(x, \omega_i) \cos \theta_i \, d\omega_i \\
\approx \frac{2\pi}{N} \sum_{i=1}^{N} L_d(p, \omega_i) \cos \theta_i
\]
Incoming direct light

\[\int_{\Omega} L_d(x, \omega_i) \cos \theta_i \, d\omega_i\]

\[\approx \frac{2\pi}{N} \sum_{i=1}^{N} L_d(p, \Omega_i) \cos \theta_i\]

\[= \int_{A..B} L_d(x, \omega_i) \cos \theta_i \, d\omega_i + \int_{C..D} L_d(x, \omega_i) \cos \theta_i \, d\omega_i\]
Incoming **direct + indirect** light
Incoming \textbf{direct} + \textbf{indirect} light

\[ -\frac{1}{2}\pi \quad +\frac{1}{2}\pi \]
Next Event Estimation

Observation: light travelling via any vertex on the path consists of indirect light and direct light *for that vertex.*

*Next Event Estimation:* sampling direct and indirect separately.
Next Event Estimation

Per surface interaction, we trace *two* random rays.

- Ray A returns (via point $x$) the energy reflected by $y$ (estimates indirect light for $x$).
- Ray B returns the direct illumination on point $x$ (estimates direct light on $x$).
- Ray C returns the direct illumination on point $y$, which will reach the sensor via ray A.
- Ray D leaves the scene.
Next Event Estimation

When a ray for indirect illumination stumbles upon a light, the path is terminated and no energy is transported via ray D:

This way, we prevent accounting for direct illumination on point y twice.
Next Event Estimation

We thus split the hemisphere into two distinct areas:

1. The area that has the projection of the light source on it;
2. The area that is not covered by this projection.

We can now safely send a ray to each of these areas and sum whatever we find there.

(or: we integrate over these non-overlapping areas and sum the energy we receive via both to determine the energy we receive over the entire hemisphere)

Area 1:
Send a ray directly to a random light source. Reject it if it hits anything else than the targeted light.

Area 2:
Send a ray in a random direction on the hemisphere. Reject it if it hits a light source.
Area 1:
Send a ray directly to a random light source. Reject it if it hits anything else than the targeted light.

Area 2:
Send a ray in a random direction on the hemisphere. Reject it if it hits a light source.
Area 1:
Send a ray directly to a random light source. Reject it if it hits anything else than the targeted light.

Area 2:
Send a ray in a random direction on the hemisphere. Reject it if it hits a light source.
Next Event Estimation

Color Sample( Ray ray )
{
    // trace ray
    I, N, material = Trace( ray );
    BRDF = material.albedo / PI;

    // terminate if ray left the scene
    if (ray.NOHIT) return BLACK;

    // terminate if we hit a light source
    if (material.isLight) return BLACK;

    // sample a random light source
    L, Nl, dist, A = RandomPointOnLight();
    Ray lr( I, L, dist );
    if (N-L > 0 && N-La-L > 0) if (!Trace( lr ))
    {
        solidAngle = ((N-La-L) * A) / dist2;
        Ld = lightColor * solidAngle * BRDF * N-L;
    }

    // continue random walk
    R = DiffuseReflection( N );
    Ray r( I, R );
    Ei = Sample( r ) * (N-R);
    return PI * 2.0f * BRDF * Ei + Ld;
}
Next Event Estimation

Some vertices require special attention:

- If the first vertex after the camera is emissive, its energy can’t be reflected to the camera.
- For specular surfaces, the BRDF to a light is always 0.

Since a light ray doesn’t make sense for specular vertices, we will include emission from a vertex directly following a specular vertex.

The same goes for the first vertex after the camera: if this is emissive, we will also include this.

*This means we need to keep track of the type of the previous vertex during the random walk.*
```cpp
Color Sample( Ray ray, bool lastSpecular )
{
    // trace ray
    I, N, material = Trace( ray );
    BRDF = material.albedo / PI;
    // terminate if ray left the scene
    if (ray.NOHIT) return BLACK;
    // terminate if we hit a light source
    if (material.isLight)
    {
        if (lastSpecular) return material.emissive;
        else return BLACK;
    }
    // sample a random light source
    L, NL, dist, A = RandomPointOnLight();
    Ray lr( I, L, dist );
    if (N.L > 0 && NL--L > 0) if (!Trace( lr ))
    {
        solidAngle = ((NL--L) * A) / dist2;
        Ld = lightColor * solidAngle * BRDF * N.L;
    }
    // continue random walk
    R = DiffuseReflection( N );
    Ray r( I, R );
    Ei = Sample( r, false ) * (N.R);
    return PI * 2.0f * BRDF * Ei + Ld;
}
```

Today's Agenda:

- Introduction
- Stratification
- Next Event Estimation
- Importance Sampling
- Multiple Importance
- Resampled Importance
Importance Sampling

Importance Sampling for Monte Carlo

Monte Carlo integration:

\[ V_A = \int_A^B f(x) \, dx = (B - A) E(f(X)) \approx \frac{B - A}{N} \sum_{i=1}^{N} f(X) \]

Example 1: rolling two dice \( D_1 \) and \( D_2 \), the outcome is \( 6D_1 + D_2 \). What is the expected value of this experiment?
(average die value is 3.5, so the answer is 3.5 * 6 + 3.5 = 24.5)

Using Monte Carlo:

\[ V = \frac{1}{N} \sum_{i=1}^{N} f(D_1) + g(D_2) \text{ where: } D_1, D_2 \in \{1,2,3,4,5,6\}, f(x) = 6x, g(x) = x \]
Importance Sampling for Monte Carlo

Changing the experiment slightly: each sample is one roll of one die.

Using Monte Carlo:

\[ V = \frac{1}{N} \sum_{i=1}^{N} f(T, D) \]

where: \( D \in \{1,2,3,4,5,6\} \), \( T \in \{0,1\} \), \( f(t, d) = (5t + 1) d \)

\[ 0.5: \text{Probability of using die } T. \]
Importance Sampling

Importance Sampling for Monte Carlo

What happens when we don't pick each die with the same probability?

```cpp
float D1_prob = 0.8f;
for( int i = 0; i < 1000; i++ )
{
    int D = IRand( 6 ) + 1;
    float r = Rand(); // uniform 0..1
    int T = (r < D1_prob) ? 0 : 1;
    float p = (T == 0) ? D1_prob : (1 - D1_prob);
    float f = (float)((5 * T + 1) * D) / p;
    total += f;
    rolls++;
}
```

- we get the correct answer;
- we get lower variance.
Importance Sampling

Importance Sampling for Monte Carlo

Example 2: sampling two area lights.

Sampling the large light with a greater probability yields a better estimate.
Importance Sampling for Monte Carlo

Example 3: sampling an integral.

Considering the previous experiments, which stratum should be sample more often?
Importance Sampling

Importance Sampling for Monte Carlo

Example 3: sampling an integral.

Considering the previous experiments, which stratum should be sampled more often?
Importance Sampling

Importance Sampling for Monte Carlo

Example 3: sampling an integral.

Considering the previous experiments, which stratum should be sample more often?

When using 8 strata and a uniform random distribution, each stratum will be sampled with a 0.125 probability. When using 8 strata and a non-uniform sampling scheme, the sum of the sampling probabilities must be 1. Good sampling probabilities are obtained by simply following the function we’re sampling. Note: we must normalize.

We don’t have to use these probabilities; any set of non-zero probabilities will work, but with greater variance. This includes any approximation of the function we’re sampling, whether this approximation is good or not.
Importance Sampling

Importance Sampling for Monte Carlo

Example 3: sampling an integral.

Considering the previous experiments, which stratum should be sample more often?

If we go from 8 to infinite strata, the probability of sampling a stratum becomes 0.

This is where we introduce the PDF, or \textit{probability density function}. On a continuous domain, the probability of sampling a specific \(X\) is 0 (just like radiance arriving at a point is 0).

However, we can say something about the probability of choosing \(X\) in a part of the domain, by integrating the pdf over the subdomain. The pdf is a \textit{probability density}.
Importance Sampling

Importance Sampling for Monte Carlo

Example 4: sampling the hemisphere.
Importance Sampling for Monte Carlo

Example 4: sampling the hemisphere.
Importance Sampling

Importance Sampling for Monte Carlo

Monte Carlo without importance sampling:

\[
E(f(X)) \approx \frac{1}{N} \sum_{i=1}^{N} f(X_i)
\]

With importance sampling:

\[
E(f(X)) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}
\]

Here, \(p(x)\) is the probability density function (PDF).
Importance Sampling

Probability Density Function

Properties of a valid PDF $p(x)$:

1. $p(x) > 0$ for all $x \in D$ where $f(x) \neq 0$

2. $\int_D p(x) d\mu(x) = 1$

*Note: $p(x)$ is a density, not a probability; it can (and will) exceed 1 for some $x$.*

Applied to direct light sampling:

$p(x) = C$ for the part of the hemisphere covered by the light source

$C = 1 / \text{solid angle to ensure } p(x) \text{ integrates to 1}$

$\Rightarrow$ Since samples are divided by $p(x)$, we multiply by $1/(1/\text{solid angle}))$:

$$L_o(p, \omega_i) \approx \text{lights} * \frac{1}{N} \sum_{i=1}^{N} f_r(p, \omega_o, P) L'_o(p, P) V(p \leftrightarrow P) \frac{A_i L_i \cos \theta_i \cos \theta_o}{\| p - P \| ^2}$$
Importance Sampling

Probability Density Function

Applied to hemisphere sampling:

Light arriving over the hemisphere is cosine weighted.

Without further knowledge of the environment, the ideal PDF is the cosine function.

$$PDF: \ p(\theta) = \cos \theta$$

Question: how do we normalize this?

$$\int_{\Omega} \cos \theta d\theta = \pi \Rightarrow \int_{\Omega} \frac{\cos \theta}{\pi} d\theta = 1$$

Question: how do we choose random directions using this PDF?
Cosine-weighted Random Direction

Without deriving this in detail:

A cosine-weighted random distribution is obtained by generating points on the unit disc, and projecting the disc on the unit hemisphere. In code:

```c
float3 CosineWeightedDiffuseReflection()
{
    float r0 = Rand(), r1 = Rand();
    float r = sqrt( r0 );
    float theta = 2 * PI * r1;
    float x = r * cosf( theta );
    float y = r * sinf( theta );
    return float3( x, y, sqrt( 1 - r0 ) );
}
```

Note: you still have to transform this to tangent space.
Importance Sampling

**Color Sample( Ray ray )**

```cpp
{
  // trace ray
  I, N, material = Trace( ray );
  // terminate if ray left the scene
  if (ray.NOHIT) return BLACK;
  // terminate if we hit a light source
  if (material.isLight) return emittance;
  // continue in random direction
  R = DiffuseReflection( N );
  Ray r( I, R );
  // update throughput
  BRDF = material.albedo / PI;
  PDF = 1 / (2 * PI);
  Ei = Sample( r ) * (N∙R) / PDF;
  return BRDF * Ei;
}
```

**Color Sample( Ray ray )**

```cpp
{
  // trace ray
  I, N, material = Trace( ray );
  // terminate if ray left the scene
  if (ray.NOHIT) return BLACK;
  // terminate if we hit a light source
  if (material.isLight) return emittance;
  // continue in random direction
  R = CosineWeightedDiffuseReflection( N );
  Ray r( I, R );
  // update throughput
  BRDF = material.albedo / PI;
  PDF = (N∙R) / PI;
  Ei = Sample( r ) * (N∙R) / PDF;
  return BRDF * Ei;
}
```
Today's Agenda:

- Introduction
- Stratification
- Next Event Estimation
- Importance Sampling
- Resampled Importance
Color Sample( Ray ray )
{
    // trace ray
    I, N, material = Trace( ray );
    BRDF = material.albedo / PI;
    // terminate if ray left the scene
    if (ray.NOHIT) return BLACK;
    // terminate if we hit a light source
    if (material.isLight) return BLACK;
    // sample a random light source
    L, NL, dist, A = RandomPointOnLight();
    Ray lr( I, L, dist );
    if (N∙L > 0 && NL∙-L > 0) if (!Trace( lr ))
    {
        solidAngle = ((NL∙-L) * A) / dist
            lightPDF = 1 / solidAngle
            E += T * (N∙L / lightPDF) * BRDF * lightColor;
    }
    // continue random walk
    R = DiffuseReflection( N );
    Ray r( I, R );
    Ei = Sample( r ) * (N-R);
    return PI * 2.0f * BRDF * Ei + Ld;
}

Color Sample( Ray ray )
{
    T = ( 1, 1, 1 ), E = ( 0, 0, 0 );
    while (1)
    {
        I, N, material = Trace( ray );
        BRDF = material.albedo / PI;
        if (ray.NOHIT) break;
        if (material.isLight) break;
        // sample a random light source
        L, NL, dist, A = RandomPointOnLight();
        Ray lr( I, L, dist );
        if (N∙L > 0 && NL∙-L > 0) if (!Trace( lr ))
        {
            solidAngle = ((NL∙-L) * A) / dist
            lightPDF = 1 / solidAngle;
            E += T * (N-L / lightPDF) * BRDF * lightColor;
        }
        // continue random walk
        R = DiffuseReflection( N );
        hemiPDF = 1 / (PI * 2.0f);
        ray = Ray( I, R );
        T *= ((N-R) / hemiPDF) * BRDF;
    }
    return E;
}
MIS

sampling the lights

\[ L_o(x, \omega_i) \approx \text{lights} \sum_{i=1}^{N} \frac{1}{N} f_r(x, \omega_o, P) L_d^I(x, P) V(x \leftrightarrow P) \frac{A_{L_d} \cos \theta_i \cos \theta_o}{\| x - P \|^2} \]

sampling the hemisphere

\[ L_o(x, \omega_o) \approx \frac{2\pi}{N} \sum_{i=1}^{N} f_r(x, \omega_o, \Omega_i) L_d(x, \Omega_i) \cos \theta_i \frac{p(\Omega_i)}{p(\Omega_i)} \]

(almost) specular

glossy

diffuse
Multiple Importance Sampling

**Light sampling**: paths to random points on the light yield high variance.

**Hemisphere sampling** (with importance): random rays yield low variance.

A ray to a random point on the light can be right in the 'peak' area of the lobe, but also in a much dimmer area. We thus get very different samples.

Here, rays sampled proportional to the BRDF have a high probability of hitting the light. Most samples yield similar energy levels.
Multiple Importance Sampling

**Light sampling:** paths to random points on the light yield low variance.

**Hemisphere sampling (with importance):** random rays yield very high variance.

*Most of these rays miss the light source.*
Multiple Importance Sampling

MIS: *combining samples taken using multiple strategies.*

1. At point $p$, we encounter a light source via light sampling: here,
   \[
   pdf(\omega) = \frac{1}{\text{solid angle}}.
   \]

2. We could have found the light source by sampling the BRDF: in that case, the pdf would have been
   \[
   pdf(\omega) = \frac{1}{2\pi}.
   \]

Since both methods are equivalent (they sample the same direction $\omega$), we can simply average the pdfs:

\[
pdf(\omega) = w p_1 + w p_2, \quad w = 0.5
\]

*(note that averaging two pdfs yields a valid pdf)*
Multiple Importance Sampling

MIS: combining samples taken using multiple strategies.

Computing optimal weights for multiple importance sampling:

$$w_i(x) = \frac{p_i(x)}{p_1(x) + p_2(x)}$$

This is known as the balance heuristic.

Note that the balance heuristic simply assigns a greater weight to the pdf with the largest value at $x$. 
Multiple Importance Sampling

Earlier, we separated direct and indirect illumination:

- Direct illumination is sampled using a PDF for a light source:
  \( p_{\text{light}}(x) = \frac{1}{\text{solidangle}} \), where \( x \) is a direction towards the light.

- Indirect illumination is sampled using a PDF proportional to the BRDF:
  \( p_{\text{brdf}}(x) = \frac{1}{2\pi} \), where \( x \) is a uniform random direction on the hemisphere, or
  \( p_{\text{brdf}}(x) = \frac{(N \cdot R)}{\pi} \), where \( x \) is a random direction using a cosine weighted distribution.

When sampling the light, we can calculate \( p_{\text{brdf}}(x) \) for that same direction.
Likewise, when we encounter a light via the BRDF pdf, we can calculate \( p_{\text{light}}(x) \).

Using these quantities, we can now balance the pdfs.
Multiple Importance Sampling

When sampling the light, we can calculate \( \pdf_{\text{brdf}}(x) \) for that same direction. Likewise, when we encounter a light via the BRDF pdf, we can calculate \( \pdf_{\text{light}}(x) \).

Example: next event estimation:

```c
L, Nl, dist, A = RandomPointOnLight();
Ray lr(I, L, dist);
if (N\cdot L > 0 && Nl\cdot -L > 0)
    if (!Trace(lr))
        {
            solidAngle = ((Nl\cdot -L) * A) / dist^2;
            lightPDF = 1 / solidAngle;
            brdfPDF = 1 / (2 * PI); // or: (N\cdot L) / PI
            misPDF = lightPDF + brdfPDF;
            E += T * (N\cdot L / misPDF) * BRDF * lightColor;
        }
```

Note: PBR (and many others) uses a different formulation, using weights. This is the same as what we are doing here. PBR says: we sample function \( f \) using 2 random directions \( x \) and \( y \), each picked from a pdf. Then:

\[
E = \frac{f(x)w_{\text{brdf}}}{p_{\text{brdf}}(x) + p_{\text{light}}(y)} + \frac{f(y)w_{\text{light}}}{p_{\text{brdf}}(x) + p_{\text{light}}(y)}
\]

where \( w_{\text{brdf}} + w_{\text{light}} = 1 \) to compensate for the fact that we are now taking two samples. Using the balance heuristic,

\[
\begin{align*}
W_{\text{brdf}} &= \frac{p_{\text{brdf}}(x)}{p_{\text{brdf}}(x) + p_{\text{light}}(y)} \\
W_{\text{light}} &= \frac{p_{\text{light}}(y)}{p_{\text{brdf}}(x) + p_{\text{light}}(y)}
\end{align*}
\]

Then, \( E = \frac{f(x)}{W_{\text{brdf}}(x) + W_{\text{light}}(y)} + \frac{f(y)}{W_{\text{brdf}}(x) + W_{\text{light}}(y)} \); this can be reduced to

\[
E = \frac{f(x)}{W_{\text{brdf}}(x)} + \frac{f(y)}{W_{\text{brdf}}(x)};
\]

Sampling \( f(x) \) alone (when stumbling upon a light) thus yields

\[
E = \frac{f(x)}{W_{\text{brdf}}(x) + W_{\text{light}}(y)};
\]

Sampling \( f(y) \) alone (next event estimation) yields

\[
E = \frac{f(y)}{W_{\text{brdf}}(x) + W_{\text{light}}(y)};
\]
Advanced Graphics – Variance Reduction

MIS

```
x = (depth < MAXDEPTH)
x = inside ? T - 1 : 0
if (n < N)
x = (r + (1 - r) * x) * n;
if (T = 1.0)
x = (0.5 * n - 0.5) * n;
if (I "diffuse"
  = true;
if (fl > refr) && (depth < MAXDEPTH)
x = (r, N);
if (refl "G" "diffuse"
  = true;

(MAXDEPTH)

survive = $SurvivalProbability: diffuse
estimation = doing it properly, (1991)

 Yazı = SampleLight: brand, I, RL, ELIGHT
x + radius.y + radius.x > 0) && (color =
  = true;
if (brdf = EvaluateDiffuse( L, N ) "Psurvive
factor = diffuse = 1.0/2;
weight = M12( directwere, brand ),
weight = M12( directwere, brand )
E = ((weight * cosThetaW) / directwere) * (radius

Window walk = done properly, closely following Schemes
Driver)
```

```
Today’s Agenda:

- Introduction
- Stratification
- Next Event Estimation
- Importance Sampling
- Resampled Importance
Many Lights
Many Lights

From Multiple to Many Lights

Potential contribution is proportional to:

- Solid angle
- Brightness of the light

Sadly we cannot precalculate potential contribution; it depends on the location and orientation of the light source relative to the point we are shading.

We can precalculate a less refined potential contribution based on:

- Area
- Brightness
Many Lights

Many Lights Array

The light array stores pointers to (or indices of) the lights in the scene. For $N$ lights, light array size $M$ is several times $N$.

Each light occupies a number of consecutive slots in the light array, proportional to its coarse potential contribution.

Selecting a random slot in the light array now yields (in constant time) a single light source $L$, with a probability of $\frac{\text{slots for } L}{M}$.
Many Lights

Resampled Importance Sampling

The light array allows us to pick a light source proportional to importance. However, this importance is not very accurate.

We can improve our choice using *resampled importance sampling.*

1. Pick \( N \) lights from the light array (where \( N \) is a small number, e.g. 4);
2. For each of these lights, determine the more accurate potential contribution;
3. Translate the potential contribution to importance;
4. Choose a light with a probability proportional to this importance.

This scheme allows for unbiased, accurate and constant time selection of a good light source.
Many Lights

Resampled Importance Sampling

Final probability for the chosen light $L$:

$$I_{\text{coarse}} \times I_{\text{resampled}}$$

Where

$$I_{\text{coarse}} = \frac{\text{slots for } L}{M} \quad \text{and} \quad I_{\text{resampled}} = \frac{\text{potential contribution } L}{\text{summed potential contributions}}.$$
INFOMAGR – Advanced Graphics
Jacco Bikker - November 2019 - February 2019

END of “Variance Reduction”
next lecture: “Various”