Ray Tracing for Games

Dr. Jacco Bikker - IGAD/BUAS, Breda, February 6

Welcome!

10

Agenda:

- Importance Sampling
- Russian Roulette

Importance Sampling for Monte Carlo

Monte Carlo integration:

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

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:

$$V = \frac{1}{N}\sum_{i=1}^{N} f(D_1) + g(D_2)\ \ where\!: D_1, D_2 \in \{1,2,3,4,5,6\},\, f(x) = 6x,\, g(x) = x$$

Importance Sampling for Monte Carlo

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

Using Monte Carlo:

$$V = \frac{1}{N}\sum_{i=1}^{N}\frac{f(T,D)}{0.5} \quad where: D \in \{1,2,3,4,5,6\}, T \in \{0,1\}, f(t,d) = (5t+1)\,d$$

0.5: Probability of using die T.

```
for( int i = 0; i < 1000; i++ )
{
   int D1 = IRand( 6 ) + 1;
   int D2 = IRand( 6 ) + 1;
   float f = (float)(6 * D1 + D2);
   total += f;
   rolls++;
}
```

```
for( int i = 0; i < 2000; i++ )
{
   int D = IRand( 6 ) + 1;
   int T = IRand( 2 );
   float f = (float)((5 * T + 1) * D) / 0.5f;
   total += f;
   rolls++;
}
```

Importance Sampling for Monte Carlo

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

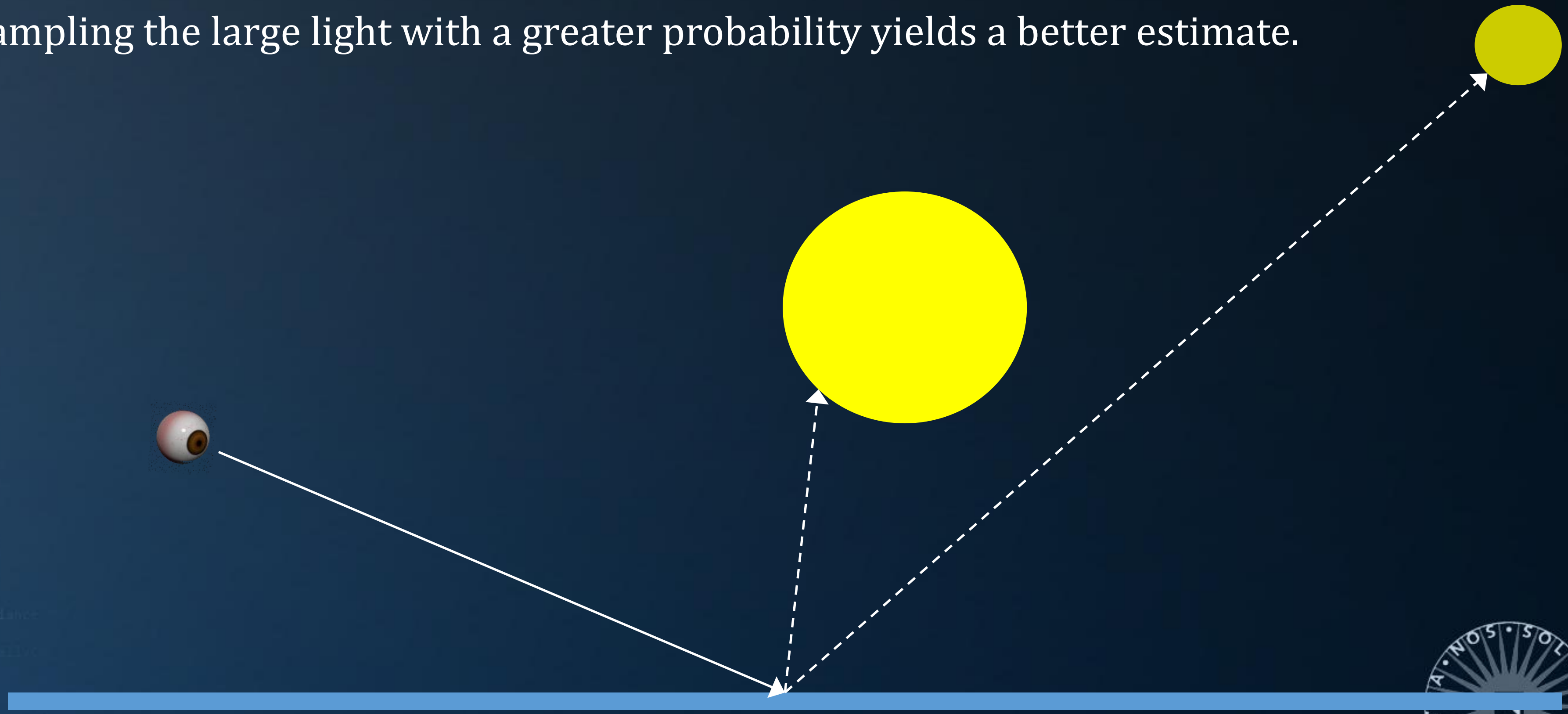
```
float D1_prob = 0.8f;

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 for Monte Carlo

Example 2: sampling two area lights.

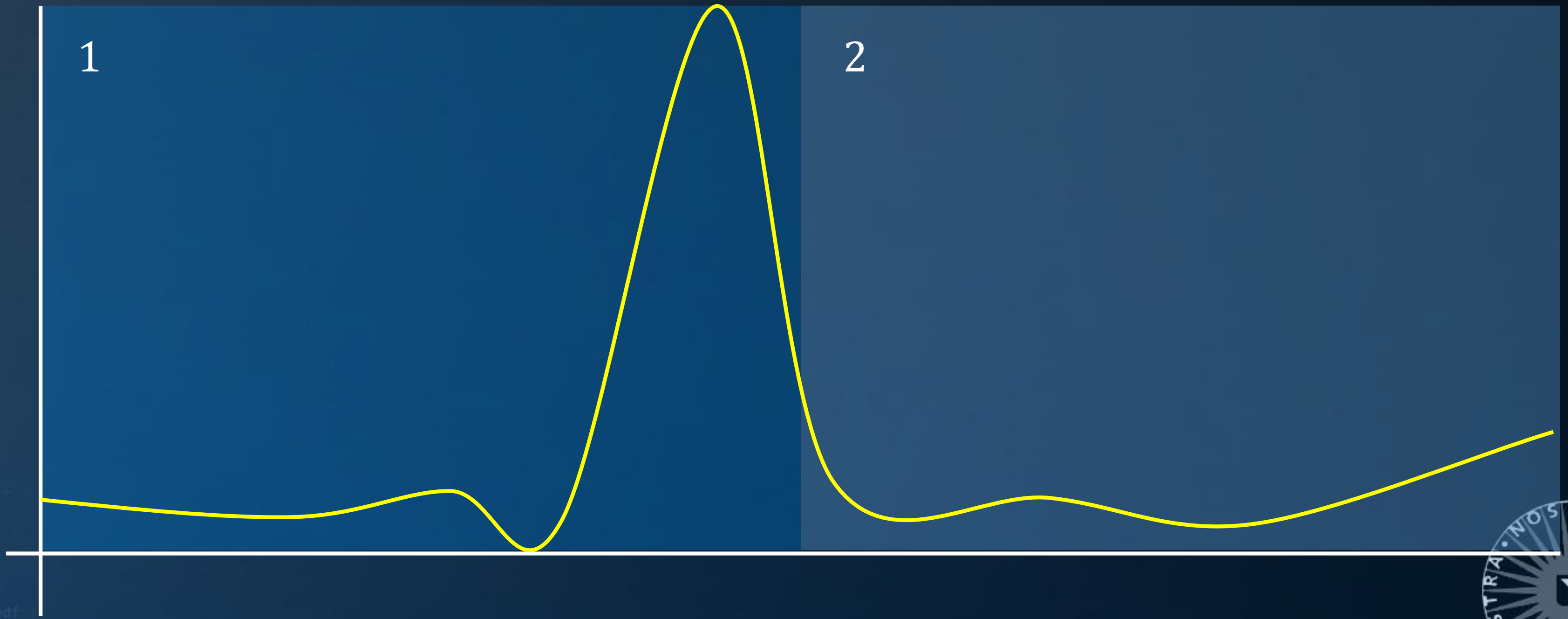
Sampling the large light with a greater probability yields a better estimate.



Importance Sampling for Monte Carlo

Example 3: sampling an integral.

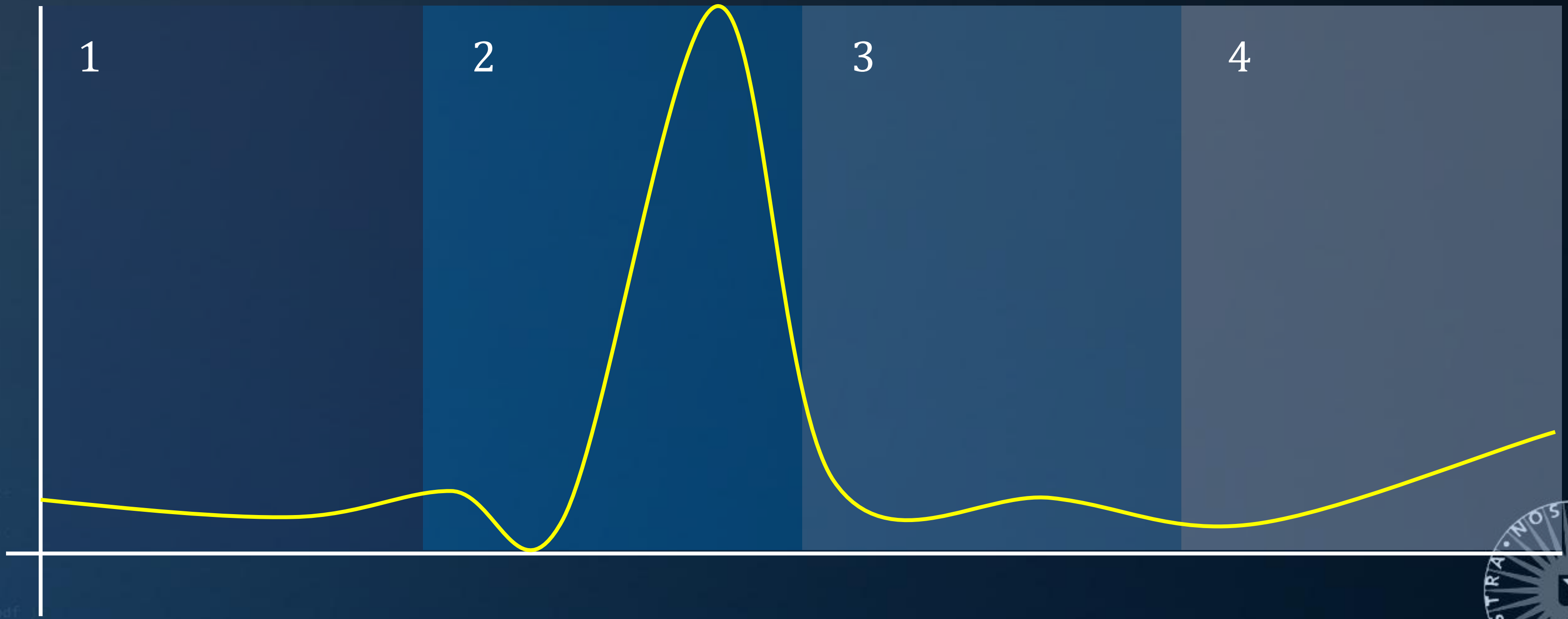
Considering the previous experiments, which stratum should be sample more often?



Importance Sampling for Monte Carlo

Example 3: sampling an integral.

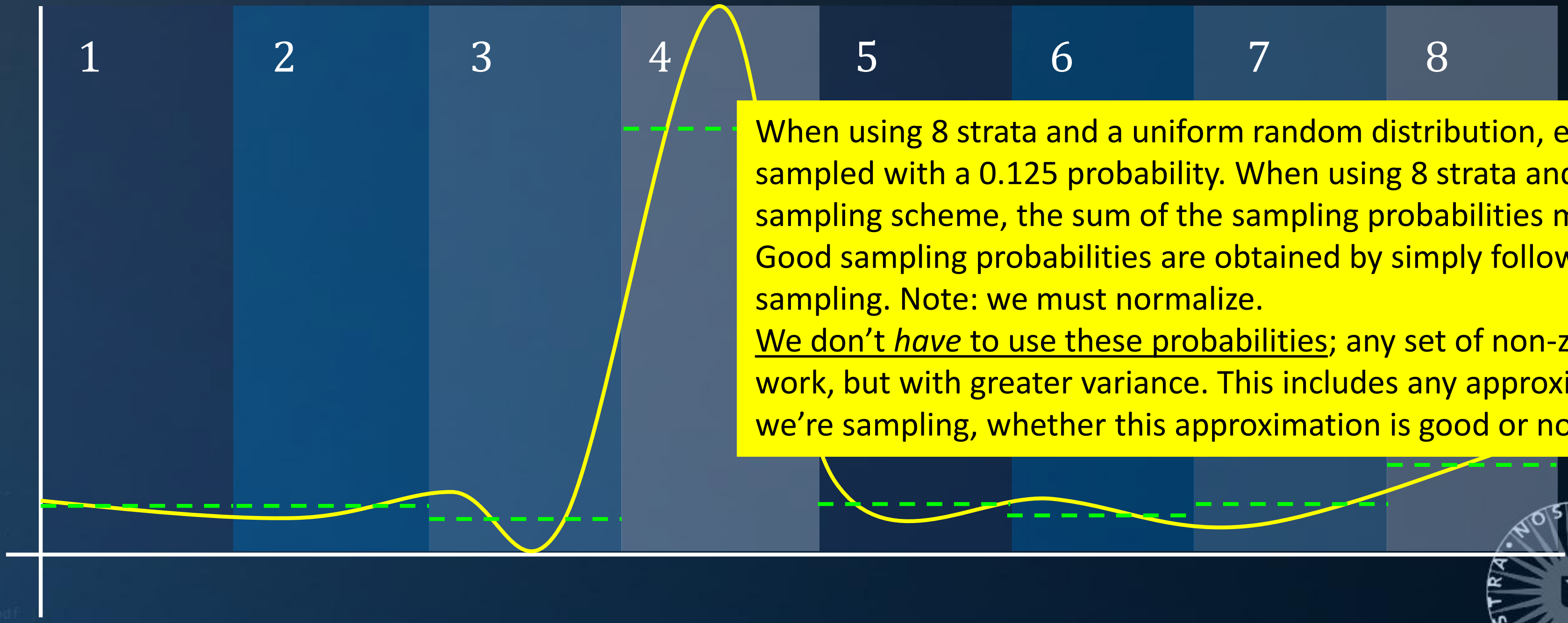
Considering the previous experiments, which stratum should be sample more often?



Importance Sampling for Monte Carlo

Example 3: sampling an integral.

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

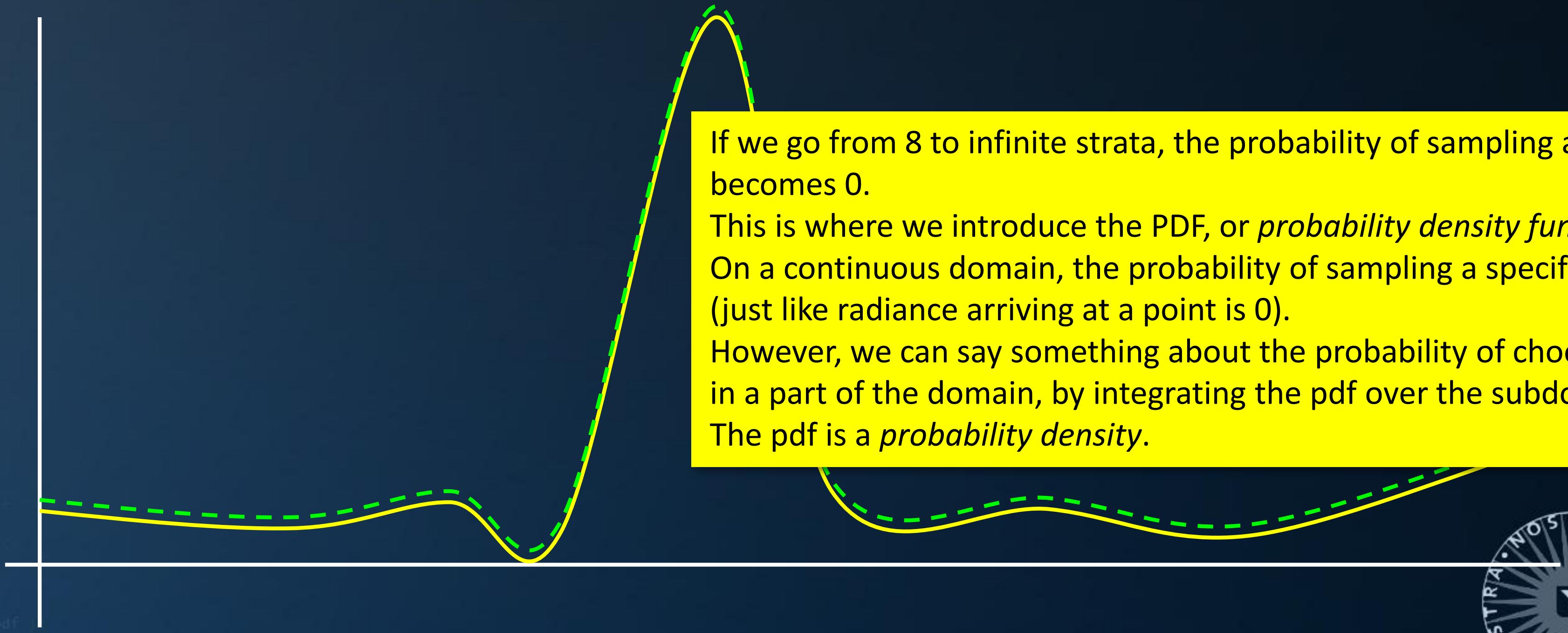


When using 8 strata and a uniform random distribution, each stratum will be sampled with a 0.125 probability. When using 8 strata and a non-uniform sampling scheme, the sum of the sampling probabilities must be 1.
Good sampling probabilities are obtained by simply following the function we're sampling. Note: we must normalize.
<u>We don't *have* to use these probabilities</u>; 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 for Monte Carlo

Example 3: sampling an integral.

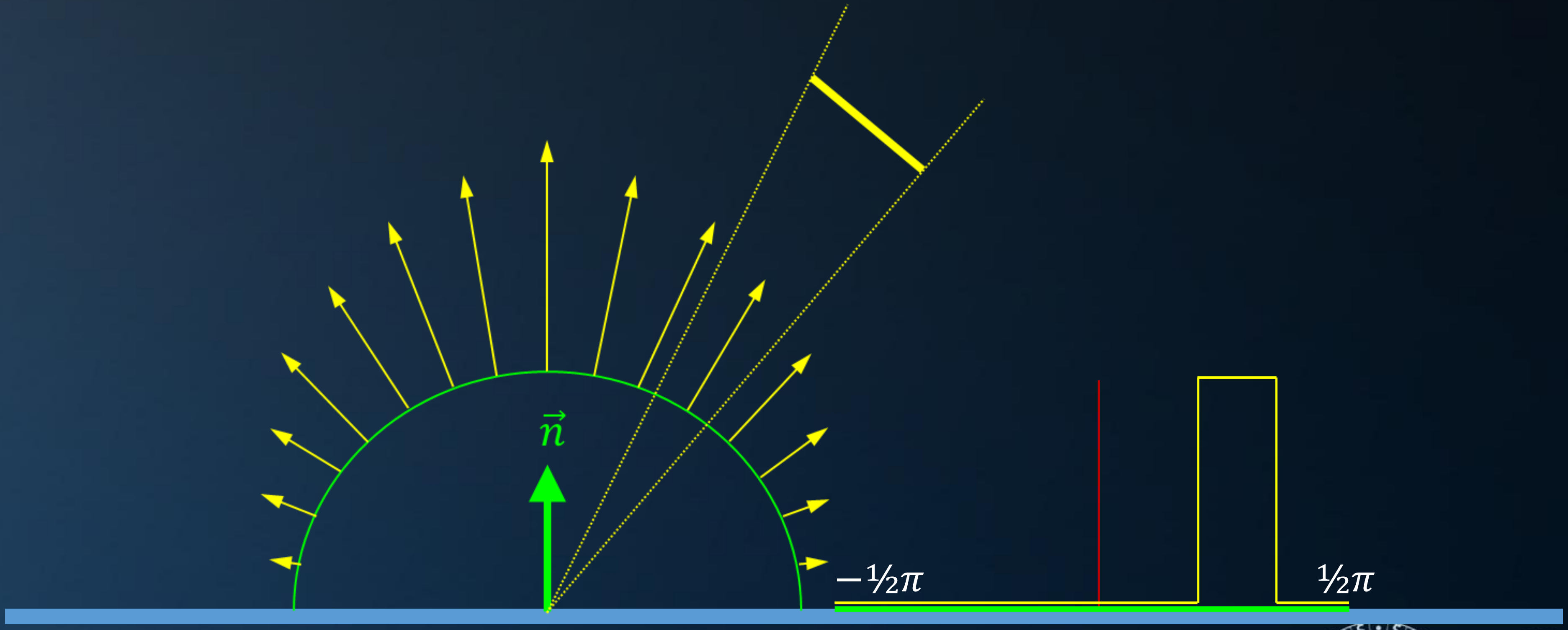
Considering the previous experiments, which stratum should be sample more often?



If we go from 8 to infinite strata, the probability of sampling a stratum becomes 0.
This is where we introduce the PDF, or *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 *probability density*.

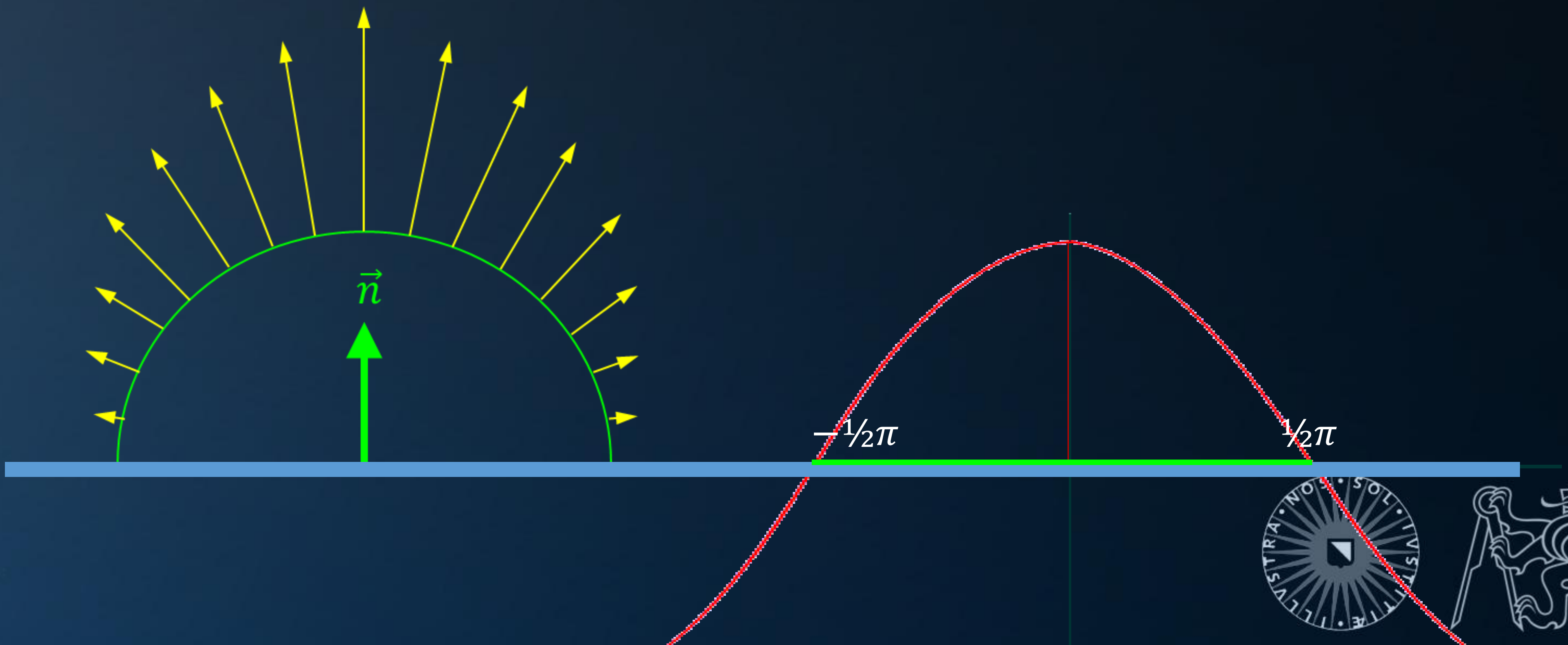
Importance Sampling for Monte Carlo

Example 4: sampling the hemisphere.



Importance Sampling for Monte Carlo

Example 4: sampling the hemisphere.



Importance Sampling for Monte Carlo

Monte Carlo without importance sampling:

$$E\big(f(X)\big) \approx \frac{1}{N}\sum_{i=1}^{N} f(X)$$

With importance sampling:

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

Here, $p(x)$ is the *probability density function* (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:

```
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.

```
Color Sample( Ray ray )
{
    // 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 )
{
    // 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;
}
```

Agenda:

- Importance Sampling
- Russian Roulette

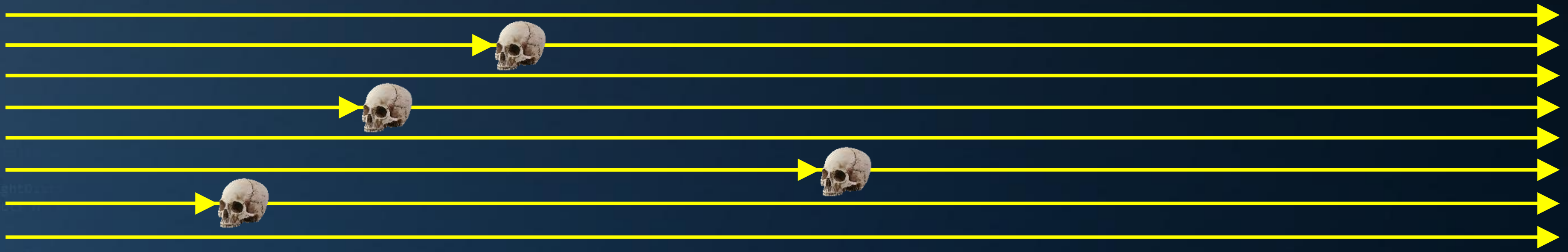
Russian Roulette

Core idea:

The longer a path becomes, the less energy it transports.

Killing half of 16 rays is easy; what do we do with a single path?

➔ Kill it with a probability of 50%.

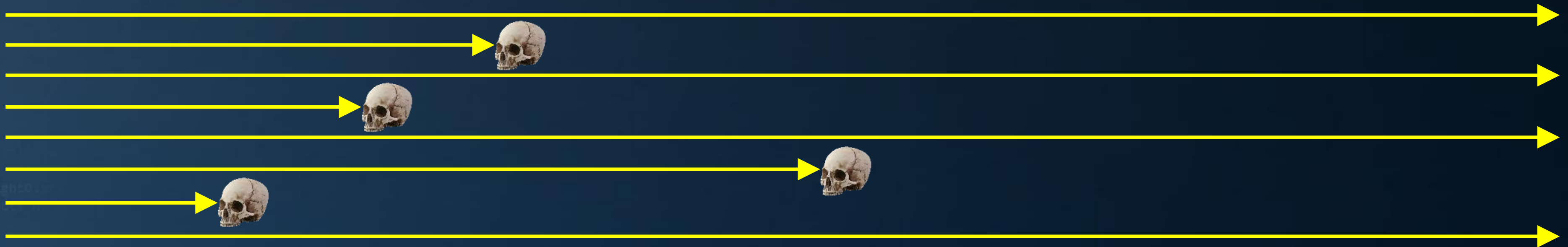


8 rays, returning 16 Watts of radiance each, 128 Watts in total.
= 4 rays, returning 32 Watts of radiance each, 128 Watts in total.

Russian Roulette

Russian roulette is applied to the random walk.

Most basic implementation: just before you start calculating the next random direction, you decide if the path lives or dies.



8 rays, returning 16 Watts of radiance each, 128 Watts in total.

= 4 rays, returning 32 Watts of radiance each, 128 Watts in total.

Better Russian Roulette

The termination probability of 50% is arbitrary.
Any probability is statistically correct.

However: for 50% survival rate, survivors scale up by 2 $\left(=\frac{1}{50\%}\right)$.
➔ In general, for a survival probability ρ, survivors scale up by $\frac{1}{\rho}$.

We can choose the survival probability *per path*. It is typically linked to albedo: the color of the last vertex. A good survival probability is:

$$\rho_{survive} = clamp\left(\frac{red + green + blue}{3}, 0.1, 0.9\right)$$

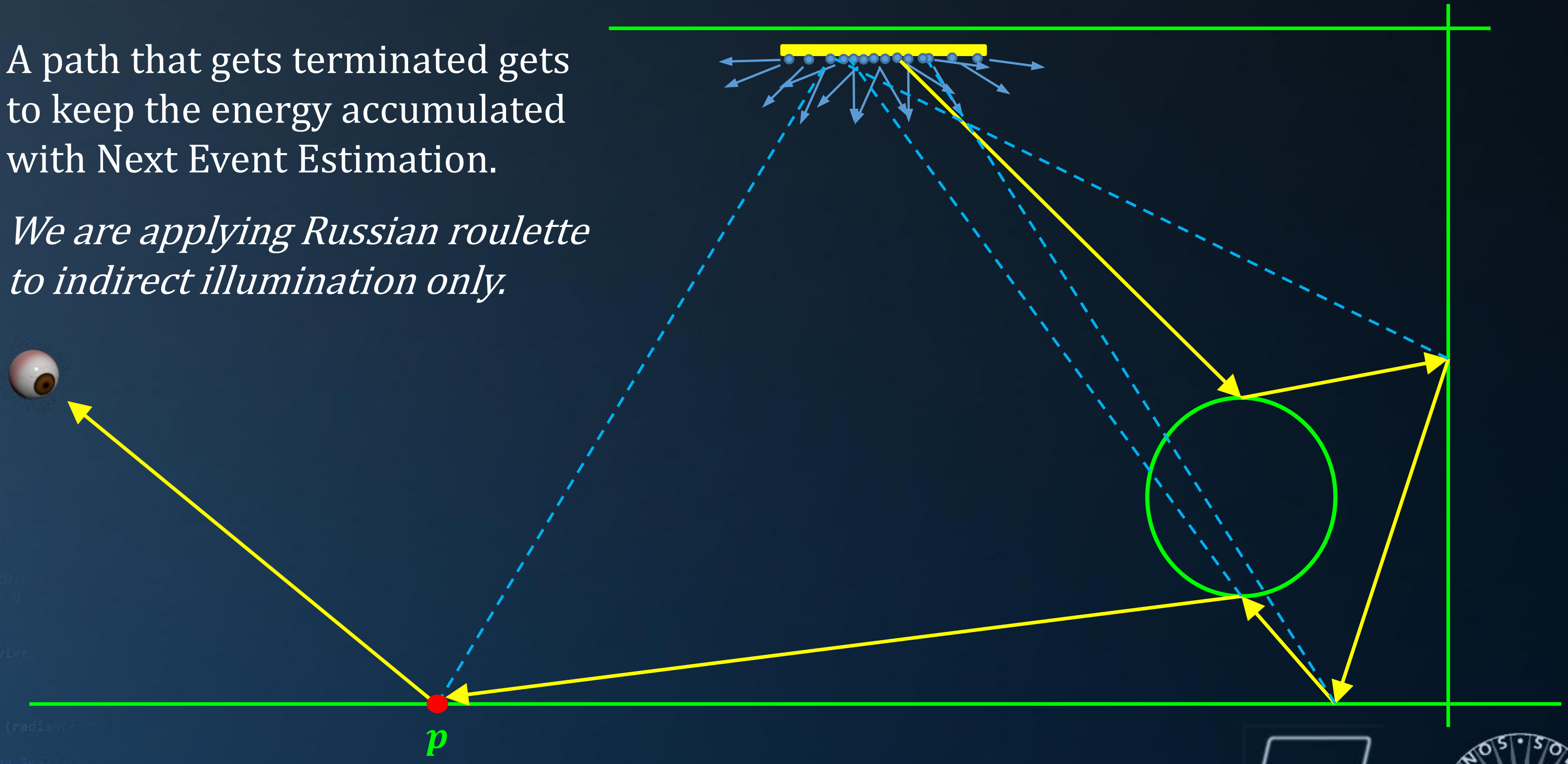
Note that $\rho_{survive} > 0$ to prevent bias. Also note that $\rho = 1$ is never a good idea.

Better:
$\rho_{survive} = clamp(\max(red, green, blue), 0, 1)$

RR and Next Event Estimation

A path that gets terminated gets to keep the energy accumulated with Next Event Estimation.

We are applying Russian roulette to indirect illumination only.



End of PART 10.