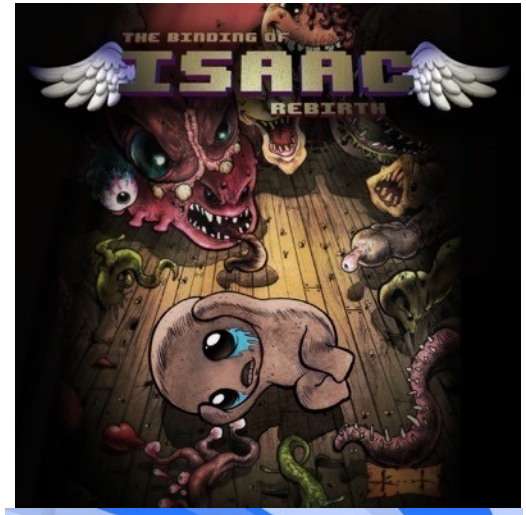
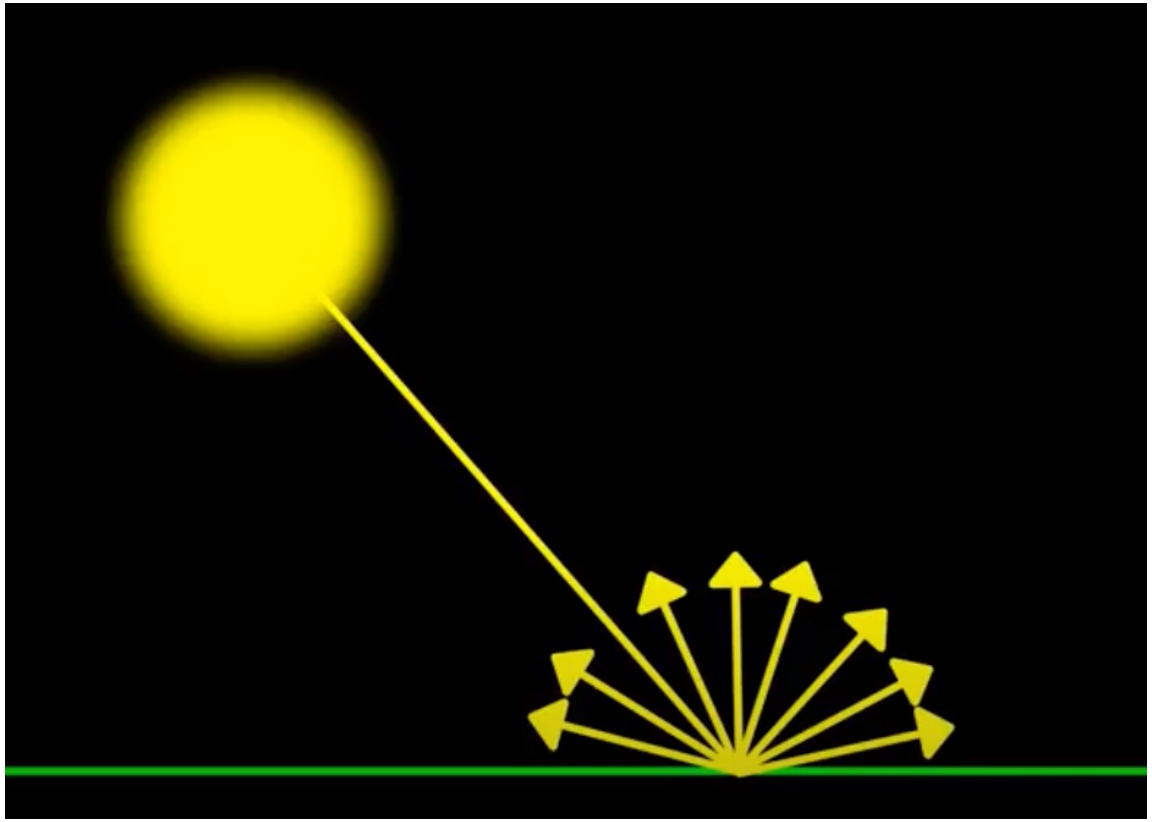
Random Numbers

Random Numbers: Applications



Random Numbers: Applications

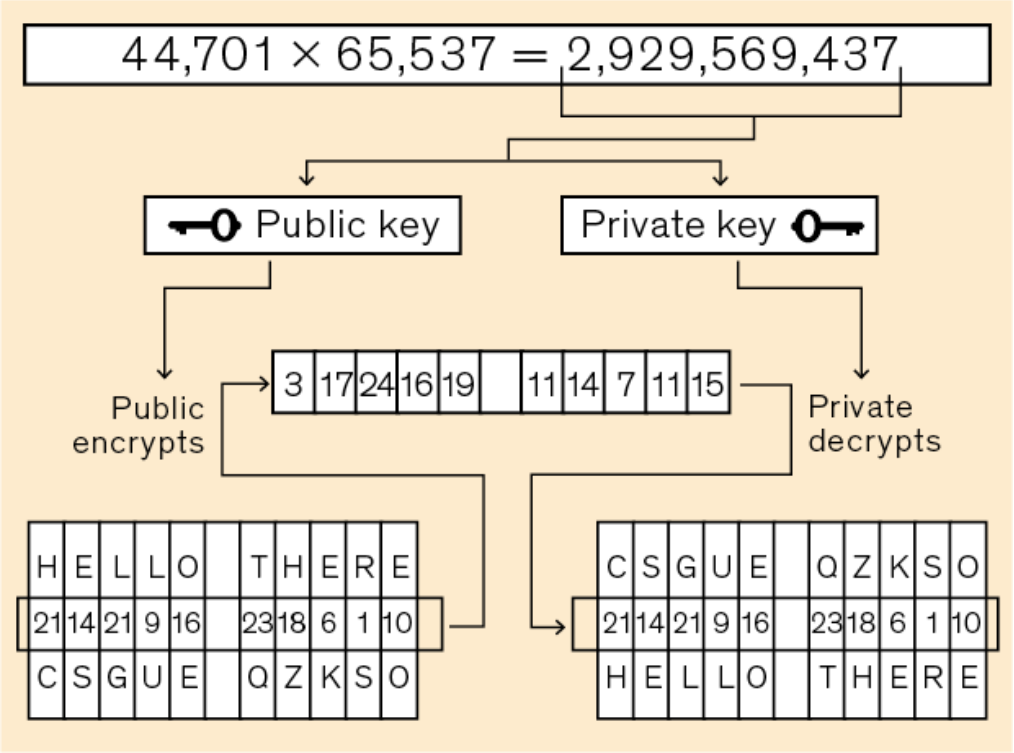
- Useful in creating **realistic graphics**
  - Cannot trace light scattering in every direction
  - Better to think of a few random directions instead
- In games that involves hitting a target, the damage caused by impact is a random number in a specified range
- Games with opponent monsters controlled by the software are a lot more fun when the monster's actions vary unpredictably



Random Numbers: Applications

- Random numbers are essential for **computer security**.
  - Every time a web connection is secured, a random number is exchanged between computers to create a temporary encryption key.

RSA Encryption Keys



Truly Random Numbers

- Difficult forcomputers to generate true random numbers
- Programs are made up of algorithms which have deterministic behavior
  - All that a computer processor can do is based on the instructions that it's given
  - Computers perform math based on the numbers stored in its memory and takes decisions based on the results of those operations.
  - Strictly following instructions cannot produce truly random numbers
  - Such process can only produce pseudo random number

A Naive Psuedo-Random Algorithm

```
value = seed
add = 11
repeat forever:
        value =value +add
        value = value % 10
        add = add+1
```

A Naive Psuedo-Random Algorithm

```
value = seed
add = 11
repeatforever:
        value =value +add
        value = value % 10
        add = add+1
```

Seed = 4

value	add
4	11
5	12
7	13
0	14
4	15
9	16
5	17
2	18

A Naive Psuedo-Random Algorithm

```
value = seed
add = 11
repeatforever:
        value =value +add
        value = value % 10
        add = add+1
```

Seed = 4

value	add
4	11
5	12
7	13
0	14
4	15
9	16
5	17
2	18

Seed = 5

value	add
5	11
6	12
8	13
1	14
5	15
0	16
6	17
3	18

A Naive Psuedo-Random Algorithm

```
value = seed
add = 11
repeatforever:
        value =value +add
        value = value % 10
        add = add+1
```

Only generates numbers between 0 and 9 and repeats after 20 values!

Seed = 4

value	add
4	11
5	12
7	13
0	14
4	15
9	16
5	17
2	18

Seed = 5

value	add
5	11
6	12
8	13
1	14
5	15
0	16
6	17
3	18

Pseudo-Random Numbers

- Generated via an algorithm
- Provided a seed value to start the generation
- Only "secure" if you don't know the algorithm or the seed
  - Also true for very complex algorithms
  - "Security" through obscurity
- Easy to generate them quickly

Pseudo-Random Numbers

- Getting different initial seeds
  - Using current time as initial seed
  - Predictable if someone knows the details of mechanism
  - Some processors have special circuits that generate truly random numbers

Generating Truly Random Numbers

- Requires entropy from the realworld
  - Entropy here is a measure of how unpredictable the information is
  - More entropy morerandom

- We can getentropyfrom:
  - User Interaction (mousemovements)
  - Atmospheric Noise
  - Radioactive Decay

When to Use

Pseudo-Random Numbers:

- Low StakesGames
- Simulations that require efficiency

Truly Random Numbers:

- Generating encryptionkeys
- High stakesgames
  - Money or tangible rewards involved
- Simulations that need true randomness

rand() & srand()

- The C standard library provides a function rand()
  - Before calling rand() we should call srand() and pass it a seed value
  - Each different seed value will result in a different pseudo-random sequence
  - srand(time(NULL)) will seed the generator with the current time
  - rand() returns a number between 0 and RAND_MAX
    - We can get random numbers within a smaller range using modular arithmetic
    - We can get random numbers between 0 and 1 with
    - rand() / (double)RAND_MAX

Rolling a die

- **roll_die()**
  - *Returns a pseudo-random number between 1 and 6.*

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int roll_die();

int main() {
    printf("rand() will return a number between 0 and %i\n", RAND_MAX);

    int seed = time(NULL);
    srand(seed);
    printf("The generator is seeded with the value %i\n", seed);

    printf("The first 10 numbers in the sequence:\n");
    for (size_t i = 0; i < 10; i++) {
        printf("%i\n", rand());
    }

    printf("Simulating rolling a die 10 times:\n");
    for (size_t i = 0; i < 10; i++) {
        printf("%i ", roll_die());
    }
    printf("\n");

    return 0;
}
```

Rolling a die

- Call srand() before calling this function.

```
int roll_die() {
    return rand() % 6 + 1;
}
```

Rolling a die

- Seeds the pseudo-random number generator used by rand() with the value seed.
- If rand() is used before any call to srand(), rand() behaves as if it was seeded with srand(1)
- Each time rand() is seeded with the same seed, it must produce the same sequence of values.
- time(NULL) gives the time as the number of seconds since January 1, 1970

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int roll_die();

int main() {
    printf("rand() will return a number between 0 and %i\n", RAND_MAX);

    int seed = time(NULL);
    srand(seed);
    printf("The generator is seeded with the value %i\n", seed);

    printf("The first 10 numbers in the sequence:\n");
    for (size_t i = 0; i < 10; i++) {
        printf("%i\n", rand());
    }

    printf("Simulating rolling a die 10 times:\n");
    for (size_t i = 0; i < 10; i++) {
        printf("%i ", roll_die());
    }
    printf("\n");

    return 0;
}
```