GTE

Advanced Topics

GTE Speedups

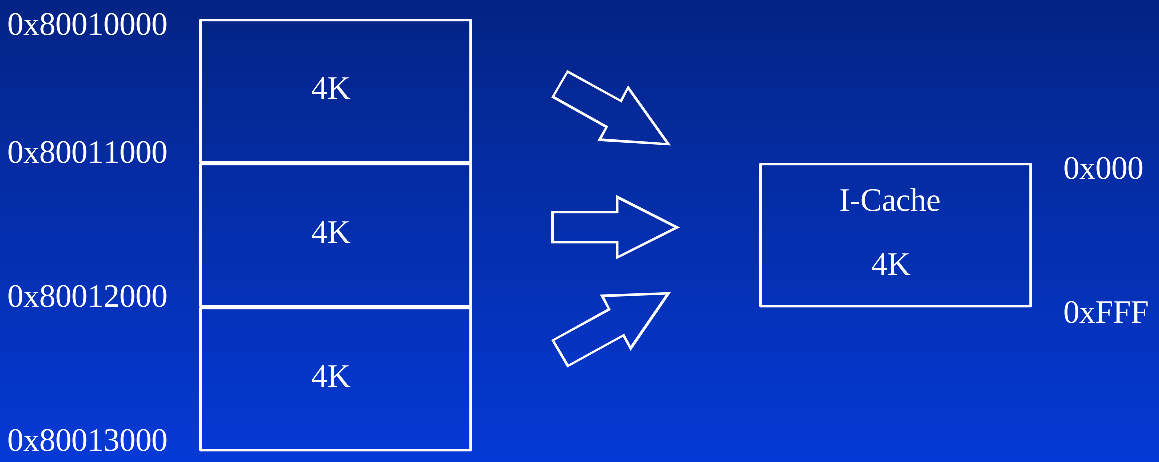
- A. Stay in I-Cache
- B. Use DMPSX
- C. Use scratchpad

A. Stay in I-Cache

1. Cache is direct mapped
2. Avoid conflicting routines
3. Stay in 4K

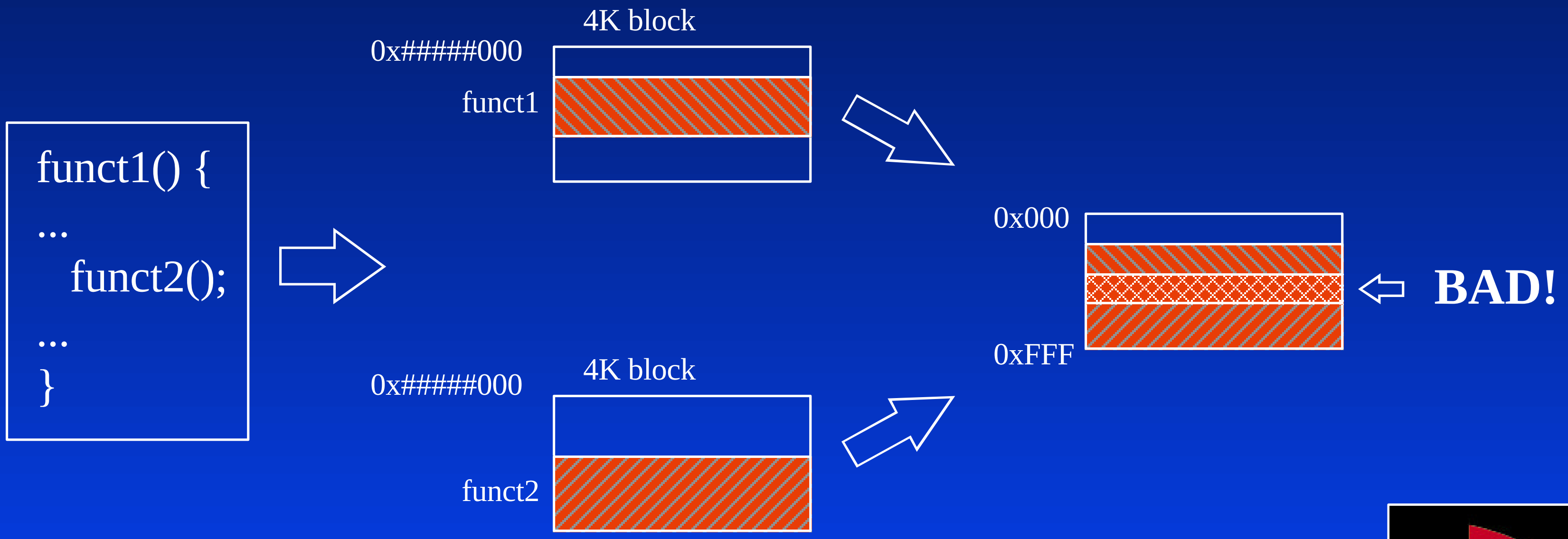
1. *I-Cache is direct mapped*

- Lower 12 bits determine placement in the cache
- Cache is loaded on 4-word boundaries



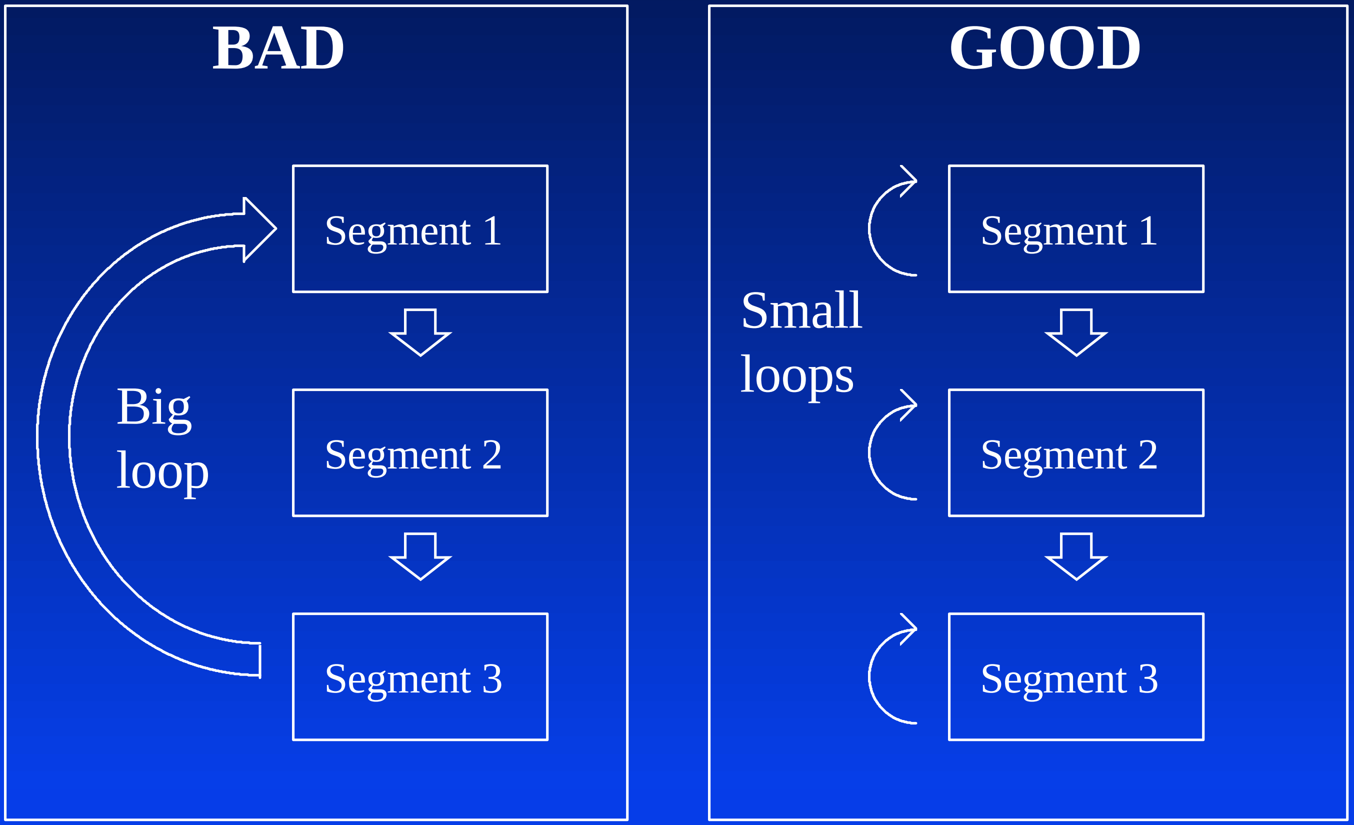
2. Avoid conflicting routines

If the lower 12 bits of two routines conflict, they will knock each other out of the I-Cache


```
funct1() {
...
   funct2();
...
}
```

3. *Stay in 4K*

Use small, tight loops to fit in the I-Cache



B. DMPSX

- What is it?

  A method for optimizing GTE commands with inline assembly macros

- What should I do with it?

  1. Eliminate unneeded GTE commands
  2. Insert R3000 commands
  3. Enable scratchpad use

PlayStation TM

Using DMPSX

GTEMAC.H = A series of replacement macros for most GTE functions

```
#define gte_RotTransPers(r1,r2,r3,r4,r5)                    \
                {               gte_ldv0(r1);               \
                                gte_rtps();                 \
                                gte_stsxy(r2);              \
                                gte_stdp(r3);               \
                                gte_stflg(r4);              \
                                gte_stszotz(r5);            }
```

INLINE.H = A set of dummy macros for subcomponents of larger macros in GTEMAC.H

```
#define gte_rtps() {\
 __asm__ volatile (".word 0x00000a3f":::"$12","$13","$14","$15","memory"); \
 __asm__ volatile (".word 0x00000a3e":::"$12","$13","$14","$15","memory"); \
 __asm__ volatile (".word 0x00000a3e":::"$12","$13","$14","$15","memory"); \
}
```

DMPSX.EXE = A post-compiler to replace dummy macros with real assembly code

1. Delete unneeded GTE commands

```
{
    gte_ldv0(v0);
    gte_rtps();
    gte_stsxy(sxy);
    gte_stdp(p);
    gte_stflg(flag);
    gte_stszotz(otz);
}
```

→

```
{
    gte_ldv0(v0);
    gte_rtps();
    gte_stsxy(sxy);
    ~~gte_stdp(p);~~
    ~~gte_stflg(flag);~~
    gte_stszotz(otz);
}
```

2. Insert R3000 commands

Three types of GTE commands

Type 1: Load GTE register	Fast
Type 2: Execute GTE instruction	Slow
Type 3: Read GTE register	Fast

Example:

```
{
    gte_ldv0(v0);          Type 1
    gte_rtps();            Type 2
    gte_stsxy(sxy);        Type 3
    gte_stszotz(otz);      Type 3
}
```

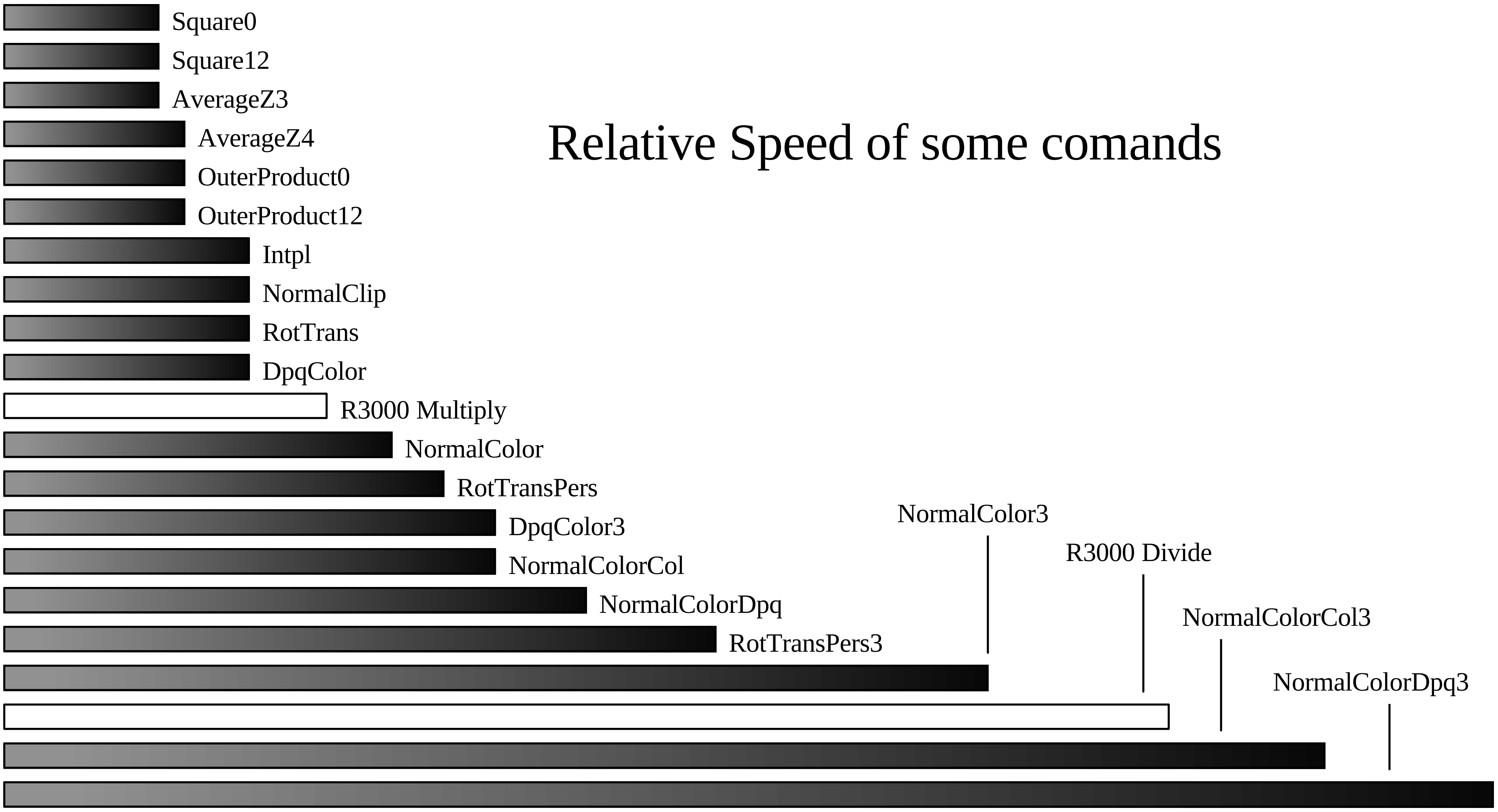
2. Insert R3000 commands (cont.)

```
{
    gte_ldv0(v0);
    gte_rtps();
    /* Type 2 = wait for GTE */
    gte_stsxy(sxy);
    gte_stszotz(otz);
}
```

→

```
{
    gte_ldv0(v0);
    gte_rtps();
    R3000 Process
    gte_stsxy(sxy);
    gte_stszotz(otz);
}
```

2. *Insert R3000 commands*



3. *Enable scratchpad use*

Slow:

- RotAverageNClip3(v0, v1, v2, sxy0, sxy1, sxy2, p, otz, flag);

4 params in Registers

5 params on stack

Fast:

gte_RotAverageNClip3_MOD();

No passed params

Which brings us to the next topic...

C. Use scratchpad

Main RAM is 5-6 X slower than Scratchpad

1. Keep local variables off stack
2. Keep passed parameters off stack
3. Use scratchpad + DMPSX
4. Put stack on scratchpad

1. Keep local variables off stack

```
typedef struct {
        u_long      *ot;
        POLY_G3 *s;
        long        otz, flg, clip;
        CVECTOR *c;
} WK;

add_cube(u_long * ot, POLY_G3 *s, SVECTOR **vp, SVECTOR **np, CVECTOR *c) {
        int         i;
        register WK *wk;

        wk = (struct wk *)getScratchAddr(0);
        wk->c = col;
        wk->ot = ot;
        for (i=0; i<12; i++,s++,vp+=3,np+=3) {
                wk->clip = RotAverageNclipColorCol3( vp[0], vp[1], vp[2],
                                                     np[0], np[1], np[2],
                                                     &(wk->c[i]),
                                                     (long *)&s->x0,(long *)&s->x1,(long *)&s->x2,
                                                     (CVECTOR *)&s->r0,(CVECTOR *)&s->r1,(CVECTOR *)&s->r2,
                                                     &wk->otz,&wk->flg);
                if (wk->clip <=0)  continue;
                if((wk->flg & 0x80000000)==0){
                        wk->otz >>= (14-OTLENGTH);
                        addPrim( wk->ot + OTSIZE - wk->otz, s);
                }}}
```

2. Keep passed params off stack

```
typedef struct {
        u_long      *ot;
        POLY_G3 *s;
        long        otz, flg, clip;
        CVECTOR *c;
} WK;

/* wk = (struct wk *)getScratchAddr(0); */                    /* set wk in calling routine */

add_cube(WK *wk, POLY_G3 *s, SVECTOR **vp, SVECTOR **np) {             /* 4 params in registers */
        int         i;

        for (i=0; i<12; i++,s++,vp+=3,np+=3) {
                wk->clip = RotAverageNclipColorCol3( vp[0], vp[1], vp[2],
                                                np[0], np[1], np[2],
                                                &(wk->c[i]),
                                                (long *)&s->x0,(long *)&s->x1,(long *)&s->x2,
                                                (CVECTOR *)&s->r0,(CVECTOR *)&s->r1,(CVECTOR *)&s->r2,
                                                &wk->otz,&wk->flg);
                if (wk->clip <=0)  continue;
                if((wk->flg & 0x80000000)==0){
                        wk->otz >>= (14-OTLENGTH);
                        addPrim( wk->ot + OTSIZE - wk->otz, s);
                }}}
```

3. Use scratchpad + DMPSX

```
add_cube(WK *wk, POLY_G3 *s, SVECTOR **vp, SVECTOR **np) {
        int         i;

        for (i=0; i<12; i++,s++,vp+=3,np+=3) {
                gte_ldv3(vp[0],vp[1],vp[2]);
                gte_rtpt();
                gte_stflg(&wk->flg);
                gte_nclip();
                gte_stopz(&wk->clip);
                if (wk->clip <= 0)  continue;
                gte_ldv3(np[0],np[1],np[2]);
                gte_ldrgb(&wk->c[i]);
                gte_ncct();
                if((wk->flg & 0x80000000)==0){
                                gte_stsxy3(&s->x0,&s->x1,&s->x2);
                                gte_strgb3(&s->r0,&s->r1,&s->r2);
                                gte_avsz3();
                                gte_stotz(&wk->otz);
                                wk->otz >>= (14-OTLENGTH);
                                addPrim( wk->ot + OTSIZE - wk->otz, s);
                }}}
```

4. Put stack on scratchpad

Only 1K!

```
/* Macros for setting stack on scratchpad */

#define SetSpadStack(addr) {\
     __asm__ volatile ("move $8,%0"
       ::"r"(addr):"$8","memory"); \
     __asm__ volatile ("sw $29,0($8)"   ::        :"$8","memory"); \
     __asm__ volatile ("addiu $8,$8,-4" ::        :"$8","memory"); \
     __asm__ volatile ("move $29,$8"    ::        :"$8","memory"); \
  }

  #define ResetSpadStack() {\
     __asm__ volatile ("addiu $29,$29,4":::"$29","memory"); \
     __asm__ volatile ("lw $29,0($29)"  :::"$29","memory"); \
  }

  #define GetStackAddr(addr) {\
     __asm__ volatile ("move $8,%0"
       ::"r"(addr):"$8","memory"); \
     __asm__ volatile ("sw $29,0($8)"   ::        :"$8","memory"); \
  }
```

```
/* sample program flow */

main()
{
        func1();
}

func1()
{
        SetSpadStack(0x1F8003FC);
        func2();
        ResetSpadStack();
}

func2();
{
        int i;

        for (i=0; i<n; i++) func3(i);
}
```

3D Troubleshooting

Cracking

Texture map distortion

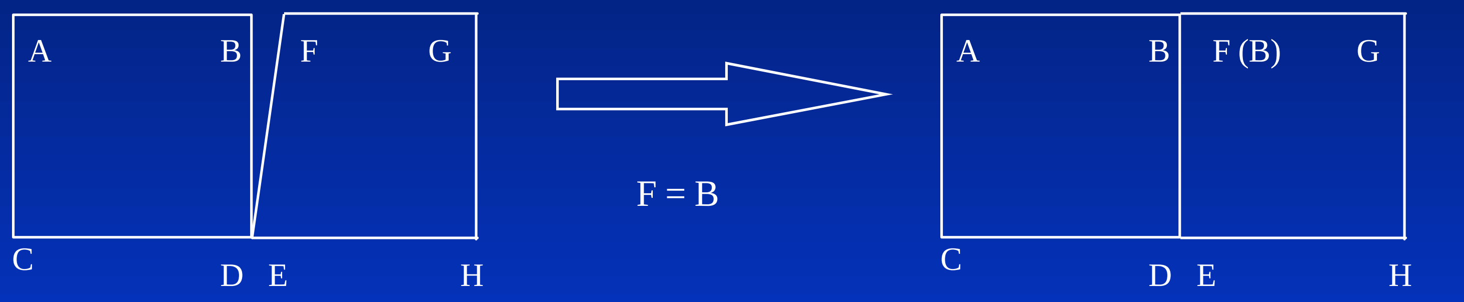
Near clip polygon dropout

Normal clip polygon dropout

Cracking

16 bit rotation inaccuracy causes cracks

Solution 1: Repair in software

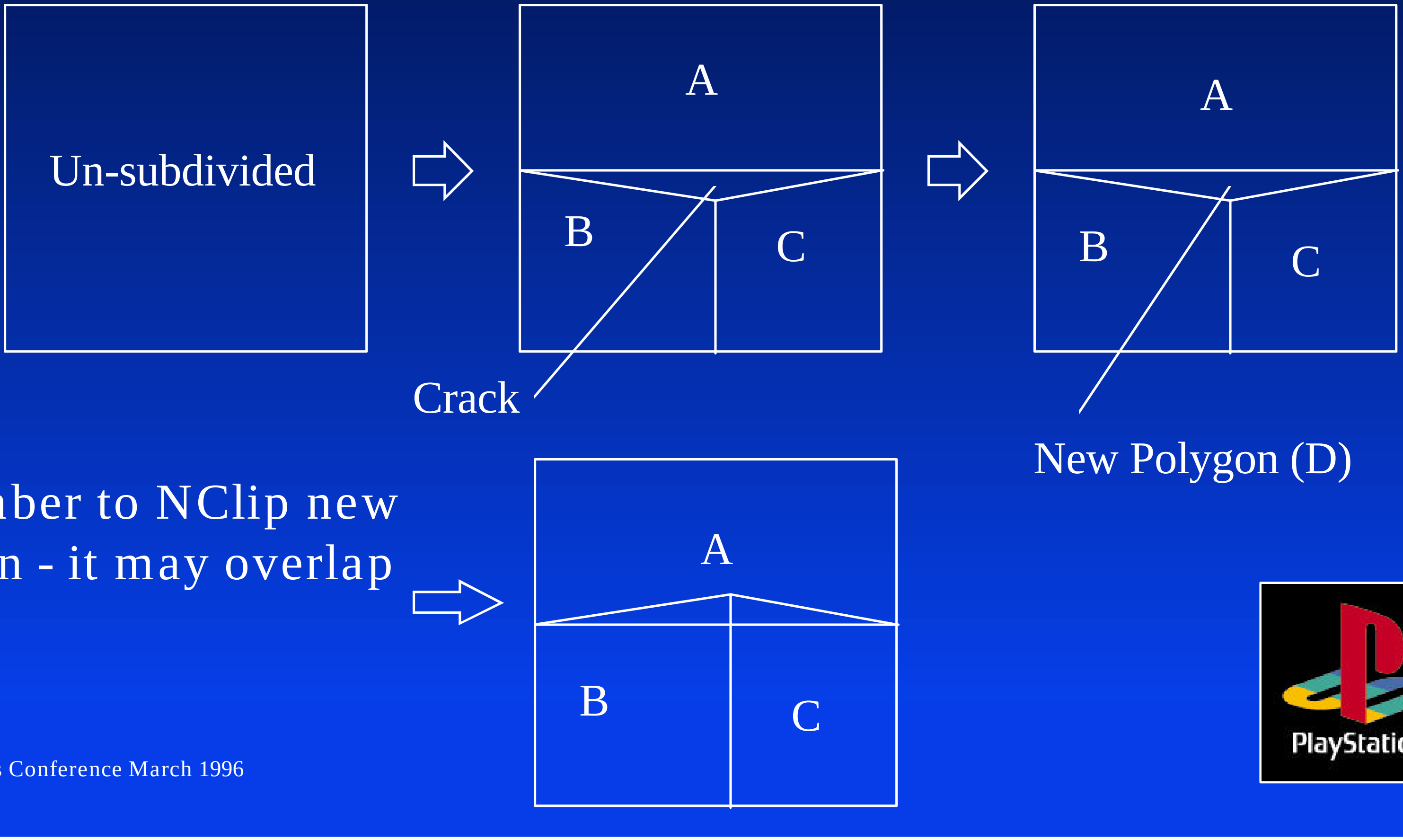


Solution 2: Use TransRot()

Cracking (cont.)

Active subdivision causes cracks

Solution 1: Add a fill poly



Remember to NClip new polygon - it may overlap

Cracking (cont.)

- General, but not-so-great solutions
  - Make polygons overlap
  - Put polygons of similar color behind regions likely to crack

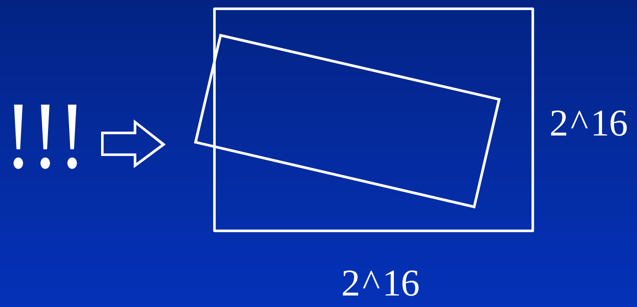
Texture map distortion

- Non-perspective texture mapping causes distortion
- Solution: subdivide polygons to reduce effect

Near clip polygon dropout

Big polygons will be clipped for two reasons

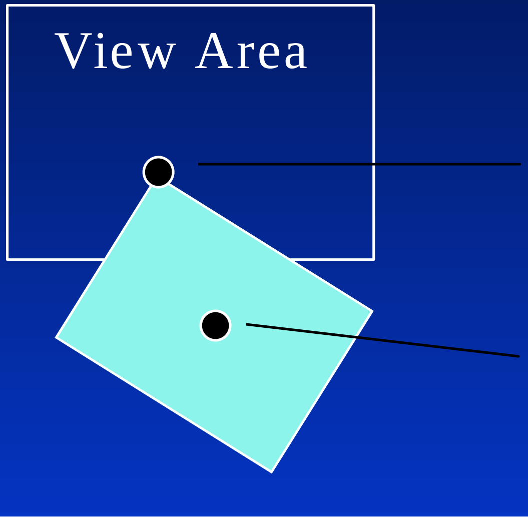
1) One endpoint out of GPU draw space



Solution: Subdivision!

Near clip polygon dropout (cont.)

2) OTZ midpoints/endpoints falling off OT



View Area

FarZ OTZ point OK
(poly will stay)

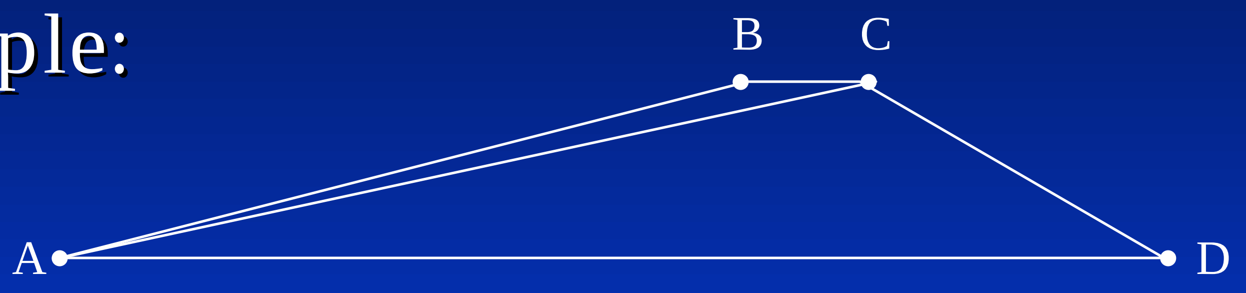
AverageZ point will be negative
(poly will miss OT)

Best solution: Subdivision!

Normal clip polygon dropout

Quads that are almost in edge view may be removed by cross-product round off error

Example:



Triangle ABC is skinny, and may fail NClip test

Solution: If first triangle fails NClip test, test second triangle as well

Note: In libgs, use funcD() calls, which means “doublecheck”

PlayStation TM

The End