Introducing the unrolled-switch anti-pattern

devblogs.microsoft.com/oldnewthing/20120403-00

April 3, 2012

Raymond Chen

Over the years, I've seen a bunch of coding anti-patterns. I figured maybe I'll share a few.

Today, I'll introduce what I'm calling the *unrolled-switch* anti-pattern, also known as "Specialization is always faster, right?"

```
enum Axis
{
    XAxis,
    YAxis,
    ZAxis,
};
// code earlier in the function ensure that
// "axis" is always a valid axis
int newPosition;
switch (axis)
{
case XAxis:
    newPosition = m_position[XAxis] + amount;
    if (newPosition < m_minPosition[XAxis])
        newPosition = m_minPosition[XAxis];
    if (newPosition > m_maxPosition[XAxis])
        newPosition = m_maxPosition[XAxis];
    m_position[XAxis] = amount;
    break;
case YAxis:
    newPosition = m_position[YAxis] + amount;
    if (newPosition < m_minPosition[YAxis])
        newPosition = m_minPosition[YAxis];
    if (newPosition > m_maxPosition[YAxis])
        newPosition = m_maxPosition[YAxis];
    m_position[YAxis] = amount;
    break;
case ZAxis:
    newPosition = m_position[ZAxis] + amount;
    if (newPosition < m_minPosition[ZAxis])
        newPosition = m_minPosition[ZAxis];
    if (newPosition > m_maxPosition[XAxis])
        newPosition = m_maxPosition[XAxis];
    m_position[ZAxis] = amount;
    break;
}
```

As we all know, special-case code is faster than general-purpose code. Instead of writing slow general-purpose code:

```
    newPosition = m_position[axis] + amount;
    if (newPosition < m_minPosition[axis])
        newPosition = m_minPosition[axis];
    if (newPosition > m_maxPosition[axis])
        newPosition = m_maxPosition[axis];
    m_position[axis] = amount;
```

we unroll it into a switch statement, thereby generating highly-optimized special-purpose code, one for each axis.

What makes this anti-pattern particularly frustrating is that you cannot tell at a glance whether all the cases really are the same (just with different axes).

In fact, they aren't.

If you look closely, you'll see that we check the new Z-position against the X-axis maximum rather than the Z-axis maximum. If you're reading this code, you now start to wonder, "Is this a copy/paste bug, or is there some reason that we really do want to check the Z-position against the X-axis minimum?"

A variation on the *unrolled-switch* is the *unrolled-if*, used if the item you want to unroll cannot be used in a *switch* statement:

```
FruitBasket *BananaBasket;
FruitBasket *AppleBasket;
FruitBasket *PearBasket;
FruitBasket *MangoBasket;
if (basket == BananaBasket) {
  if (!BananaBasket->IsEmpty()) {
    fruit = BananaBasket->TakeFruit();
    if (HaveKnife()) {
      TakeKnife();
      fruit->Peel();
      fruit->Slice();
      fruit->Eat();
      ReplaceKnife();
    } else {
      BananaBasket->AddFruit(fruit);
    }
  }
} else if (basket == AppleBasket) {
  if (!AppleBasket->IsEmpty()) {
    fruit = AppleBasket->TakeFruit();
    if (HaveKnife()) {
      TakeKnife();
      fruit->Peel();
      fruit->Slice();
      fruit->Eat();
      ReplaceKnife();
    } else {
      AppleBasket->AddFruit(fruit);
    }
  }
} else if (basket == PearBasket) {
  if (!PearBasket->IsEmpty()) {
    fruit = PearBasket->TakeFruit();
    if (HaveKnife()) {
      TakeKnife();
      fruit->Slice();
      fruit->Eat();
      ReplaceKnife();
    } else {
      PearBasket->AddFruit(fruit);
    }
  }
} else if (basket == MangoBasket) {
  if (!MangoBasket->IsEmpty()) {
    fruit = MangoBasket->TakeFruit();
    if (HaveKnife()) {
      TakeKnife();
      fruit->Peel();
      fruit->Slice();
      fruit->Eat();
      ReplaceKnife();
    } else {
      BananaBasket->AddFruit(fruit);
    }
  }
}
```

When I pointed out in an aside to the customer that this could be simplified (after fixing the copy/paste errors) to

```
if (!basket->IsEmpty()) {
  fruit = basket->TakeFruit();
  if (HaveKnife()) {
    TakeKnife();
    fruit->Peel();
    fruit->Slice();
    fruit->Eat();
    ReplaceKnife();
  } else {
    basket->AddFruit(fruit);
  }
}
```

the response was, “Hey, that’s a neat trick. I didn’t realize you could do that.”

I wonder if this person also programs loops like this:

```
switch (limit)
{
case 0:
  break;
case 1:
  do_something(array[0]);
  break;
case 2:
  for (int i = 0; i < 2; i++) do_something(array[i]);
  break;
case 3:
  for (int i = 0; i < 3; i++) do_something(array[i]);
  break;
case 4:
  for (int i = 0; i < 4; i++) do_something(array[i]);
  break;
case 5:
  for (int i = 0; i < 5; i++) do_something(array[i]);
  break;
case 6:
  for (int i = 0; i < 6; i++) do_something(array[i]);
  break;
...
case 999:
  for (int i = 0; i < 999; i++) do_something(array[i]);
  break;
default:
  FatalError("Need more cases to handle larger array");
  break;
}
```

Raymond Chen

Follow